

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of :
Taketo HEISHI et al. :
Serial No. NEW : **Attn: APPLICATION BRANCH**
Filed January 21, 2004 : Attorney Docket No. 2004_0070A

COMPILER APPARATUS AND
COMPILATION METHOD

THE COMMISSIONER IS AUTHORIZED
TO CHARGE ANY DEFICIENCY IN THE
FEES FOR THIS PAPER TO DEPOSIT
ACCOUNT NO. 23-0975

CLAIM OF PRIORITY UNDER 35 USC 119

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450


Sir:

Applicants in the above-entitled application hereby claim the date of priority under the International Convention of Japanese Patent Application No. 2003-019365, filed January 28, 2003, as acknowledged in the Declaration of this application.

A certified copy of said Japanese Patent Application is submitted herewith.

Respectfully submitted,

Taketo HEISHI et al.

By 
Michael S. Huppert
Registration No. 40,268
Attorney for Applicants

MSH/kjf
Washington, D.C. 20006-1021
Telephone (202) 721-8200
Facsimile (202) 721-8250
January 21, 2004



日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日 2 0 0 3 年 1 月 2 8 日
Date of Application:

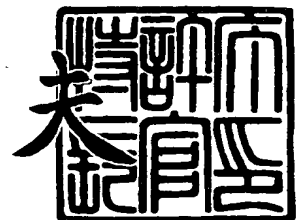
出 願 番 号 特 願 2 0 0 3 - 0 1 9 3 6 5
Application Number:
[ST. 10/C] : [J P 2 0 0 3 - 0 1 9 3 6 5]

出 願 人 松 下 電 器 産 業 株 式 会 社
Applicant(s):

2 0 0 3 年 1 0 月 1 0 日

特許庁長官
Commissioner,
Japan Patent Office

今 井 康 夫



出証番号 出証特 2 0 0 3 - 3 0 8 3 7 9 8

【書類名】 特許願

【整理番号】 5037740129

【あて先】 特許庁長官殿

【国際特許分類】 G06F 9/45

【発明者】

 【住所又は居所】 大阪府門真市大字門真 1 0 0 6 番地 松下電器産業株式会社内

 【氏名】 瓶子 岳人

【発明者】

 【住所又は居所】 大阪府門真市大字門真 1 0 0 6 番地 松下電器産業株式会社内

 【氏名】 小川 一

【発明者】

 【住所又は居所】 大阪府門真市大字門真 1 0 0 6 番地 松下電器産業株式会社内

 【氏名】 谷 丈暢

【発明者】

 【住所又は居所】 大阪府門真市大字門真 1 0 0 6 番地 松下電器産業株式会社内

 【氏名】 笹川 幸宏

【特許出願人】

 【識別番号】 000005821

 【氏名又は名称】 松下電器産業株式会社

【代理人】

 【識別番号】 100109210

 【弁理士】

 【氏名又は名称】 新居 広守

 【電話番号】 06-4806-7530

【手数料の表示】

【予納台帳番号】 049515

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 0213583

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 コンパイラ装置およびコンパイル方法

【特許請求の範囲】

【請求項 1】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令とのハミング距離が小さくなる命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置し、前記中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段と

を備えることを特徴とするコンパイラ装置。

【請求項 2】 前記最適化手段は、直前のサイクルの同一の命令発行ユニットに対応する位置に配置された命令のオペコードとのハミング距離が小さくなる命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置することを特徴とする請求項 1 に記載のコンパイラ装置。

【請求項 3】 前記最適化手段は、直前のサイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタ番号とのハミング距離が小さくなる命令を優先して、前記複数の命令発行ユニットの各々に対応する位置に当該命令を配置することを特徴とする請求項 1 に記載のコンパイラ装置。

【請求項 4】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタと同一のレジスタをアクセスする命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置し、前記中間コードを最適化する最適化手段と

最適化された前記中間コードを機械語命令に変換するコード生成手段とを備えることを特徴とするコンパイラ装置。

【請求項 5】 前記最適化手段は、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタのレジスタ番号と近い番号のレジスタをアクセスする命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置することを特徴とする請求項 4 に記載のコンパイラ装置。

【請求項 6】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、前記複数の命令発行ユニットの各々には、あらかじめ優先的に発行される命令が規定されており、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットの各々で優先発行される命令を優先して、前記複数の命令発行ユニットの各々に対応する位置に当該命令を配置し、前記中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段とを備えることを特徴とするコンパイラ装置。

【請求項 7】 ソースプログラムを、並列処理可能な複数の実行ユニットと

、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットのうち特定の命令発行ユニットに対応する位置には命令が配置されないように、前記複数の命令発行ユニットの各々の対応する位置に命令を配置し、前記中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段とを備えることを特徴とするコンパイラ装置。

【請求項 8】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記複数の命令発行ユニットにそれぞれ対応する複数の命令の配置位置について、同一数の命令の未配置位置があらかじめ定められた命令サイクル数以上連続する区間を検出する区間検出手段と、

前記区間の直前に、前記命令の未配置位置に対応する命令発行ユニットの動作を停止させるための命令を挿入する第 1 の命令挿入手段と、

当該命令が挿入された前記中間コードを機械語命令に変換するコード生成手段と

を備えることを特徴とするコンパイラ装置。

【請求項 9】 さらに、前記区間の直後に、前記命令の未配置位置に対応する命令発行ユニットの動作を復帰させるための命令を挿入する第 2 の命令挿入手

段

を備えることを特徴とする請求項 8 に記載のコンパイラ装置。

【請求項 10】 前記プロセッサは、前記複数の命令発行ユニットの利用状態を示す値を保持するプログラム状態レジスタを有し、

前記第 1 の命令挿入手段で挿入される命令は、前記プログラム状態レジスタに前記複数の命令発行ユニットの利用状態を示す値を書き込むことを特徴とする請求項 8 または 9 に記載のコンパイラ装置。

【請求項 11】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムは、前記プロセッサが使用する命令発行ユニットの個数を指定可能な個数指定情報を含み、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、前記個数指定情報で指定された個数の命令発行ユニットのみを動作させるように命令を配置し、前記中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段と

を備えることを特徴とするコンパイラ装置。

【請求項 12】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記プロセッサが使用する命令発行ユニットの個数を受付ける受け手段と、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、前記受付手段が受付けた前記個数の命令発行ユニットのみを動作させるように命令を配置し、前記中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段とを備えることを特徴とするコンパイラ装置。

【請求項 13】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令とのハミング距離が小さくなるように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段とを備えることを特徴とするコンパイラ装置。

【請求項 14】 前記最適化手段は、直前の命令サイクルの複数の命令発行ユニットに対応する位置にそれぞれ配置された複数の命令とのハミング距離の合計値が小さくなるように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行なうことを特徴とする請求項 13 に記載のコンパイラ装置。

【請求項 15】 前記最適化手段は、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のオペコードとのハミング距離が小さくなるように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行なうことを特徴とする請求項 13 に記載のコンパイラ装置。

【請求項 16】 前記最適化手段は、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタ番号とのハミング距離が小

さくなるように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行なうことを特徴とする請求項 13 に記載のコンパイラ装置。

【請求項 17】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、各命令発行ユニットにおいて、直前の命令サイクルの命令のレジスタと同一のレジスタのアクセスが最も多く連続するように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段とを備えることを特徴とするコンパイラ装置。

【請求項 18】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、前記複数の命令発行ユニットの各々には、あらかじめ優先的に発行される命令群が規定されており、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットの各々で優先的に発行される命令群に最も多く合致するように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段と
を備えることを特徴とするコンパイラ装置。

【請求項 19】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットに対応する位置に命令を配置する命令スケジューリング手段と、

前記命令スケジューリング手段でスケジュールされた各命令で使用される変数について、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタ番号とのハミング距離が小さくなるようなレジスタを優先して、当該変数に割付けるレジスタ割付手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段と
を備えることを特徴とするコンパイラ装置。

【請求項 20】 前記レジスタ割付手段は、スケジュールされた各命令で使用される変数について、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令と同一のレジスタを優先して、当該変数に割付けることを特徴とする請求項 19 に記載のコンパイラ装置。

【請求項 21】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、

前記ソースプログラムを構文解析するパーサー手段と、

解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、

前記中間コードに対応する命令の依存関係を崩すことなく、複数のあらかじめ

定められた方法の各々に従い、前記複数の命令発行ユニットの各々の対応する位置に命令を配置し、生成された複数の配置命令のうち、プログラム実行時に最も消費電力が小さくなると予想される配置命令を選択し、前期中間コードを最適化する最適化手段と、

最適化された前記中間コードを機械語命令に変換するコード生成手段とを備えることを特徴とするコンパイラ装置。

【請求項 22】 前記最適化手段は、

前記中間コードに対応する命令の依存関係を崩すことなく、複数のあらかじめ定められた配置方法の各々に従い、前記複数の命令発行ユニットに対応する位置に命令を配置する命令スケジューリング部と、

前記命令スケジューリング手段でスケジュールされた各命令で使用される変数について、複数のあらかじめ定められた割付方法の各々に従い、レジスタを前記変数に割付けるレジスタ割付部とを有し、

前記命令スケジューリング部での配置方法と、前記レジスタ割付部での割付方法とを、プログラム実行時に最も消費電力が小さくなると予想される命令の配置が得られるように、バックトラック方式で探索する

ことを特徴とする、請求項 21 に記載のコンパイラ装置。

【請求項 23】 前記最適化手段は、複数のあらかじめ定められた再配置方法の各々に従い、前記複数の命令発行ユニットに対応する位置に、前記レジスタ割付部で変数にレジスタが割付けられた命令を、再配置する命令再スケジューリング部をさらに有し、

前記命令スケジューリング部での配置方法と、前記レジスタ割付部での割付方法と、前記命令再スケジューリング部での再配置方法とを、プログラム実行時に最も消費電力が小さくなると予想される命令の配置が得られるように、バックトラック方式で探索する

ことを特徴とする、請求項 22 に記載のコンパイラ装置。

【請求項 24】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法で

あって、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令とのハミング距離が小さくなる命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするコンパイル方法。

【請求項 25】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタと同一のレジスタをアクセスする命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするコンパイル方法。

【請求項 26】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、前記複数の命令発行ユニットの各々には、あらかじめ優先的に発行される命令が規定されており、

前記ソースプログラムを構文解析するパーサーステップと、
解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットの各々で優先発行される命令を優先して、前記複数の命令発行ユニットの各々に対応する位置に当該命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするコンパイル方法。

【請求項 27】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記ソースプログラムを構文解析するパーサーステップと、
解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットのうち特定の命令発行ユニットに対応する位置には命令が配置されないように、前記複数の命令発行ユニットの各々の対応する位置に命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするコンパイル方法。

【請求項 28】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記ソースプログラムを構文解析するパーサーステップと、
解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記複数の命令発行ユニットにそれぞれ対応する複数の命令の配置位置について、同一数の命令の未配置位置があらかじめ定められた命令サイクル数以上連続する区間を検出する区間検出ステップと、

前記区間の直前に、前記命令の未配置位置に対応する命令発行ユニットの動作を停止させるための命令を挿入する第1の命令挿入ステップと、

当該命令が挿入された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするコンパイル方法。

【請求項29】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記ソースプログラムは、前記プロセッサが使用する命令発行ユニットの個数を指定可能な個数指定情報を含み、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記個数指定情報で指定された個数の命令発行ユニットのみを動作させるように命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするコンパイル方法。

【請求項30】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記プロセッサが使用する命令発行ユニットの個数を受付ける受け付けステップと、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記個数の命令発行ユニットのみを動作させるように命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするコンパイル方法。

【請求項 3 1】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令とのハミング距離が小さくなるように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするコンパイル方法。

【請求項 3 2】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、各命令発行ユニッ

トにおいて、直前の命令サイクルの命令のレジスタと同一のレジスタのアクセスが最も多く連続するように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするコンパイル方法。

【請求項 33】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、前記複数の命令発行ユニットの各々には、あらかじめ優先的に発行される命令群が規定されており、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットの各々で優先的に発行される命令群に最も多く合致するように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするコンパイル方法。

【請求項 34】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットに対応する位置に命令を配置する命令スケジューリングステップと、

スケジュールされた各命令で使用される変数について、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタ番号とのハミング距離が小さくなるようなレジスタを優先して、当該変数に割付けるレジスタ割付ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするコンパイル方法。

【請求項 35】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイル方法であって、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、複数のあらかじめ定められた方法の各々に従い、前記複数の命令発行ユニットの各々の対応する位置に命令を配置し、生成された複数の配置命令のうち、プログラム実行時に最も消費電力が小さくなると予想される配置命令を選択し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするコンパイル方法。

【請求項 36】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイク

ルの同一の命令発行ユニットに対応する位置に配置された命令とのハミング距離が小さくなる命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするプログラム。

【請求項 37】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタと同一のレジスタをアクセスする命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするプログラム。

【請求項 38】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、前記複数の命令発行ユニットの各々には、あらかじめ優先的に発行される命令が規定されており、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットの各々で優先発行される命令を優先して、前記複数の命令発行ユニットの各々に対応する位置に当該命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするプログラム。

【請求項 39】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットのうち特定の命令発行ユニットに対応する位置には命令が配置されないように、前記複数の命令発行ユニットの各々の対応する位置に命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするプログラム。

【請求項 40】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記複数の命令発行ユニットにそれぞれ対応する複数の命令の配置位置につい

て、同一数の命令の未配置位置があらかじめ定められた命令サイクル数以上連続する区間を検出する区間検出ステップと、

前記区間の直前に、前記命令の未配置位置に対応する命令発行ユニットの動作を停止させるための命令を挿入する第 1 の命令挿入ステップと、

当該命令が挿入された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするプログラム。

【請求項 4 1】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記ソースプログラムは、前記プロセッサが使用する命令発行ユニットの個数を指定可能な個数指定情報を含み、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記個数指定情報で指定された個数の命令発行ユニットのみを動作させるように命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップと

を含むことを特徴とするプログラム。

【請求項 4 2】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記プロセッサが使用する命令発行ユニットの個数を受付ける受け付けステップと、

前記ソースプログラムを構文解析するパーサーステップと、
解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記個数の命令発行ユニットのみを動作させるように命令を配置し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするプログラム。

【請求項 4 3】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、
解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令とのハミング距離が小さくなるように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするプログラム。

【請求項 4 4】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、各命令発行ユニットにおいて、直前の命令サイクルの命令のレジスタと同一のレジスタのアクセスが最も多く連続するように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするプログラム。

【請求項 4 5】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、前記複数の命令発行ユニットの各々には、あらかじめ優先的に発行される命令群が規定されており、

前記プログラムは、

前記ソースプログラムを構文解析するパーサーステップと、

解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットの各々で優先的に発行される命令群に最も多く合致するように、着目している命令サイクルに配置された命令の当該命令サイクル内での並べ替えを行ない、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするプログラム。

【請求項 4 6】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記ソースプログラムを構文解析するパーサステップと、
解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットに対応する位置に命令を配置する命令スケジューリングステップと、
スケジュールされた各命令で使用される変数について、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタ番号とのハミング距離が小さくなるようなレジスタを優先して、当該変数に割付けるレジスタ割付ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするプログラム。

【請求項 47】 ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ用のプログラムであって、

前記プログラムは、

前記ソースプログラムを構文解析するパーサステップと、
解析された前記ソースプログラムを中間コードに変換する中間コード変換ステップと、

前記中間コードに対応する命令の依存関係を崩すことなく、複数のあらかじめ定められた方法の各々に従い、前記複数の命令発行ユニットの各々の対応する位置に命令を配置し、生成された複数の配置命令のうち、プログラム実行時に最も消費電力が小さくなると予想される配置命令を選択し、前記中間コードを最適化する最適化ステップと、

最適化された前記中間コードを機械語命令に変換するコード生成ステップとを含むことを特徴とするプログラム。

【請求項 48】 並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語コードが記録されたコンピュータ読取可能な記録媒体であって

前記機械語コードは、

前記複数の命令発行ユニットのいずれかで、あらかじめ定められた命令サイクル数以上命令が実行されない期間において当該命令発行ユニットの動作を停止させる第 1 の命令

を備えることを特徴とするコンピュータ読取可能な記録媒体。

【請求項 49】 前記機械語コードは、さらに、

前記期間の経過直後に当該命令発行ユニットの動作を復帰させるための第 2 の命令

を備えることを特徴とする請求項 48 に記載のコンピュータ読取可能な記録媒体。

【請求項 50】 前記プロセッサは、前記複数の命令発行ユニットの利用状態を示す値を保持するプログラム状態レジスタを有し、

前記第 1 の命令は、前記プログラム状態レジスタに前記複数の命令発行ユニットの利用状態を示す値を書き込むことを特徴とする請求項 48 または 49 に記載のコンピュータ読取可能な記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、C/C++言語等の高級言語で記述されたソースプログラムを機械語プログラムに変換するコンパイラに関し、特に、プログラム実行時の消費電力が少ない機械語プログラムを出力可能なコンパイラに関する。

【0002】

【従来の技術】

近年普及している携帯電話、情報携帯端末等の携帯型の情報処理装置では、消費電力の低減が要求されている。このため、このような情報処理装置で利用されるプロセッサが備える高い機能を有効に引き出しつつ、低消費電力で動作可能な機械語命令を生成可能なコンパイラが求められている。

【0003】

従来のコンパイラとして、命令の実行順序を変更することによりプロセッサの消費電力の低減を図った命令列最適化装置がある（たとえば、特許文献1参照）。

【0004】

この命令列最適化装置では、命令の依存関係に影響を与えることなく、命令のビットパターン間のハミング距離を低減させるように命令の配置変更を行なう。これにより、プロセッサの消費電力を低減させることができる命令列の最適化を行なうことができる。

【0005】

【特許文献1】

特開平8-101777号公報

【0006】

【発明が解決しようとする課題】

しかしながら、従来の命令列最適化装置では、並列処理可能なプロセッサを前提としていない。このため、従来の最適化処理を並列処理可能なプロセッサにそのまま適用しても、最適な命令列の最適化を得ることができないという問題がある。

【0007】

そこで、本発明は、このような状況に鑑みてなされたものであり、並列処理可能なプロセッサを低消費電力で動作させることができる命令列を生成可能なコンパイラを提供することを目的とする。

【0008】

【課題を解決するための手段】

上記目的を達成するために、本発明に係るコンパイラ装置は、ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、前記ソースプログラムを構文解析するパーサー手段と、解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、前記中間コードに対応する命令の依存関係を崩すこ

となく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令とのハミング距離が小さくなる命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置し、前記中間コードを最適化する最適化手段と、最適化された前記中間コードを機械語命令に変換するコード生成手段とを備えることを特徴とする。

【0009】

これにより、各命令発行ユニットで実行される命令のビットパターンの変化を抑えることができるので、プロセッサの命令レジスタに保持される値のビット変化が小さく、プロセッサを低消費電力で動作させることができる命令列が生成される。

【0010】

また、本発明に係るコンパイラ装置は、ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、前記ソースプログラムを構文解析するパーサー手段と、解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、前記中間コードに対応する命令の依存関係を崩すことなく、直前の命令サイクルの同一の命令発行ユニットに対応する位置に配置された命令のレジスタと同一のレジスタをアクセスする命令を優先して、前記複数の命令発行ユニットの各々の対応する位置に当該命令を配置し、前記中間コードを最適化する最適化手段と、最適化された前記中間コードを機械語命令に変換するコード生成手段とを備えることを特徴とする。

【0011】

これにより、同一のレジスタへのアクセスが連続し、レジスタを選択するための制御信号の変化が少なくなり、プロセッサを低消費電力で動作させることができる命令列が生成される。

【0012】

さらに、本発明に係るコンパイラ装置は、ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行

する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、前記複数の命令発行ユニットの各々には、あらかじめ優先的に発行される命令が規定されており、前記ソースプログラムを構文解析するパーサー手段と、解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、前記中間コードに対応する命令の依存関係を崩すことなく、前記複数の命令発行ユニットの各々で優先発行される命令を優先して、前記複数の命令発行ユニットの各々に対応する位置に当該命令を配置し、前記中間コードを最適化する最適化手段と、最適化された前記中間コードを機械語命令に変換するコード生成手段とを備えることを特徴とする。

【0013】

これにより、同じ命令発行ユニットで優先発行される命令として、プロセッサの同じ構成要素を利用する命令を割り当てれば、同一の命令発行ユニットでは、同じ構成要素を利用する命令が連続して実行されることになるため、プロセッサを低消費電力で動作させることができる命令列が生成される。

【0014】

さらにまた、本発明に係るコンパイラ装置は、ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、前記ソースプログラムを構文解析するパーサー手段と、解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、前記複数の命令発行ユニットにそれぞれ対応する複数の命令の配置位置について、同一数の命令の未配置位置があらかじめ定められた命令サイクル数以上連続する区間を検出する区間検出手段と、前記区間の直前に、前記命令の未配置位置に対応する命令発行ユニットの動作を停止させるための命令を挿入する第1の命令挿入手段と、当該命令が挿入された前記中間コードを機械語命令に変換するコード生成手段とを備えることを特徴とする。

【0015】

これにより、命令発行ユニットに対応する位置に命令が連続して配置されていない場合には、その間、その命令発行ユニットへの電力の供給を停止させること

ができたため、プロセッサを低消費電力で動作させることができる命令列が生成される。

【0016】

さらにまた、本発明に係るコンパイラ装置は、ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、前記ソースプログラムは、前記プロセッサが使用する命令発行ユニットの個数を指定可能な個数指定情報を含み、前記ソースプログラムを構文解析するパーサー手段と、解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、前記中間コードに対応する命令の依存関係を崩すことなく、前記個数指定情報で指定された個数の命令発行ユニットのみを動作させるように命令を配置し、前記中間コードを最適化する最適化手段と、最適化された前記中間コードを機械語命令に変換するコード生成手段とを備えることを特徴とする。

【0017】

これにより、個数指定情報で指定された箇所の命令に関しては、命令が供給されない命令発行ユニットを発生させることができ、その命令発行ユニットへの電力の供給を停止させることができるため、プロセッサを低消費電力で動作させることができる命令列が生成される。

【0018】

さらにまた、本発明に係るコンパイラ装置は、ソースプログラムを、並列処理可能な複数の実行ユニットと、前記複数の実行ユニットで実行される命令を各々発行する複数の命令発行ユニットとを有するプロセッサ用の機械語プログラムに翻訳するコンパイラ装置であって、前記プロセッサが使用する命令発行ユニットの個数を受付ける受け付け手段と、前記ソースプログラムを構文解析するパーサー手段と、解析された前記ソースプログラムを中間コードに変換する中間コード変換手段と、前記中間コードに対応する命令の依存関係を崩すことなく、前記受け付け手段が受け付けた前記個数の命令発行ユニットのみを動作させるように命令を配置し、前記中間コードを最適化する最適化手段と、最適化された前記中間コード

を機械語命令に変換するコード生成手段とを備えることを特徴とする。

【0019】

これにより、受付手段で受付けた個数の命令発行ユニットのみを動作させ、その他の命令発行ユニットへの電力の供給を停止させることができるため、プロセッサを低消費電力で動作させることができる命令例が生成される。

【0020】

なお、本発明は、このようなコンパイラ装置として実現することができるだけでなく、このようなプログラムに含まれる手段をステップとするコンパイル方法として実現したり、このような特徴的なコンパイラ用のプログラムまたはコンピュータ読取可能な記録媒体として実現したりすることもできる。そして、そのようなプログラムやデータファイルは、CD-ROM (Compact Disk-Read Only Memory) 等の記録媒体やインターネット等の伝送媒体を介して広く流通させることができるのは言うまでもない。

【0021】

【発明の実施の形態】

以下、本発明に係るコンパイラの実施の形態について図面を用いて詳細に説明する。

【0022】

本実施の形態におけるコンパイラは、C/C++言語等の高級言語で記述されたソースプログラムを特定のプロセッサ（ターゲット）が実行できる機械語プログラムに翻訳するクロスコンパイラであり、プロセッサの消費電力を低減させることができるという特徴を有する。

【0023】

[プロセッサ]

まず、本実施の形態におけるコンパイラの対象となるプロセッサの一例について、図1～図11を用いて説明する。

【0024】

本実施の形態におけるコンパイラの対象となるプロセッサは、例えば、通常のマイコンに比べて実行可能な命令の並列性が高く、複数の命令を並列して処理で

きるようにパイプライン方式が採用されている。

【0025】

図1は、本実施の形態に係るプロセッサが解読実行する命令の構造を示す図である。

図1(a)～図1(d)を参照して、本プロセッサの各命令は、32ビットの固定長である。各命令の0ビット目は、並列実行境界情報を示す。並列実行境界情報が“1”のときは、その命令と後続命令との間に並列実行の境界が存在し、並列実行境界情報が“0”のときは、並列実行の境界が存在しないことになる。並列実行境界情報の利用方法については、後述する。

【0026】

各命令の命令長から並列実行境界情報を除いた31ビットの部分においてオペレーションを決定する。具体的には、フィールド“Op1”、“Op2”、“Op3”および“Op4”では、オペレーションの種類を示すオペコードを指定する。レジスタフィールド“Rs”、“Rs1”および“Rs2”では、ソースオペランドとなるレジスタのレジスタ番号を指定する。レジスタフィールド“Rd”では、ディスティネーションオペランドとなるレジスタのレジスタ番号を指定する。フィールド“Imm”では、演算用定数オペランドを指定する。フィールド“Disp”では、変位（ディスプレースメント）を指定する。

【0027】

オペコードの先頭2ビット（30および31ビット目）は、オペレーションの種類（オペレーション群）を指定するために用いられる。その詳細については、後述する。

【0028】

オペコードOp2～Op4は、16ビット長のデータであるが、オペコードOp1は21ビット長のデータである。したがって、便宜的に、オペコードOp1の前半部分（16～31ビット目）をオペコードOp1-1と呼び、後半部分（11～15ビット目）をオペコードOp1-2と呼ぶ。

【0029】

図2は、本実施の形態に係るプロセッサの概略構成を示すブロック図である。

プロセッサ 30 は、VLIW (Very Long Instruction Word) 方式に従って記述された命令群 (以下「パケット」と呼ぶ。) を記憶する命令メモリ 40 と、命令供給発行部 50 と、解読部 60 と、実行部 70 と、データメモリ 100 とを含む。各部の詳細については後述する。

【0030】

図 3 は、パケットの一例を示す図である。1 パケットは、命令フェッチの単位であり、4 命令から構成されるものと定義する。上述のように、1 命令は 32 ビット長である。このため、1 パケットは、128 ($= 32 \times 4$) ビット長である。

【0031】

再度図 2 を参照して、命令供給発行部 50 は、命令メモリ 40、解読部 60 および実行部 70 に接続され、実行部 70 より供給される PC (プログラムカウンタ) の値に基づいて、命令メモリ 40 よりパケットを受信し、解読部 60 に並列して最高 3 つの命令を供給する。

【0032】

解読部 60 は、命令供給発行部 50 および実行部 70 に接続され、命令供給発行部 50 から供給された命令を解読し、実行部 70 に供給する。

実行部 70 は、命令供給発行部 50、解読部 60 およびデータメモリ 100 に接続され、解読部 60 より供給される解読結果に基づいて、必要に応じデータメモリ 100 に記憶されたデータアクセスを行ない、命令に基づいた処理を実行する。また、処理が実行されるごとに、実行部 70 は PC の値を 1 つずつインクリメントする。

【0033】

命令供給発行部 50 は、命令メモリ 40 および実行部 70 内の後述する PC 部に接続され、PC 部に保持されたプログラムカウンタで示される命令メモリ 40 のアドレスにアクセスし、命令メモリ 40 よりパケットを受信する命令フェッチ部 52 と、命令フェッチ部 52 に接続され、パケットを一時的に保持する命令バッファ 54 と、命令バッファ 54 に接続され、パケットに含まれる命令を最大 3 つ保持する命令レジスタ部 56 とを含む。

【0034】

命令フェッチ部 52 および命令メモリ 40 は、I A (Instruction Address) バス 42 および I D (Instruction Data) バス 44 により接続されている。I A バス 42 は 32 ビット幅であり、I D バス 44 は 128 ビット幅である。命令フェッチ部 52 から命令メモリ 40 へのアドレス供給は、I A バス 42 を介して行われる。命令メモリ 40 から命令フェッチ部 52 へのパケットの供給は、I D バス 44 を解して行われる。

【0035】

命令レジスタ部 56 は、各々命令バッファ 54 に接続され、それぞれ 1 つの命令を保持する命令レジスタ 56 a ~ 56 c を含む。

解読部 60 は、命令レジスタ部 56 内の 3 つの命令レジスタ 56 a ~ 56 c に保持された命令の発行に関する制御を行う命令発行制御部 62 と、命令発行制御部 62 および命令レジスタ部 56 に接続され、命令発行制御部 62 の制御に基づいて、命令レジスタ部 56 から供給される命令をデコードするデコード部 64 とを含む。

【0036】

デコード部 64 は、命令レジスタ 56 a ~ 56 c にそれぞれ接続され、基本的に 1 サイクルに 1 つの命令を解読し、制御信号を出力する命令デコーダ 64 a ~ 64 c を含む。

【0037】

実行部 70 は、デコード部 64 に接続され、デコード部 64 内の 3 つの命令デコーダ 64 a ~ 64 c より出力される制御信号に基づいて、実行部 70 内の後述する各構成要素を制御する実行制御部 72 と、次に実行すべきパケットのアドレスを保持する P C 部 74 と、32 個の 32 ビットレジスタ R 0 ~ R 31 から構成されるレジスタファイル 76 と、各々、S I M D (Single Instruction Multiple Data) 型命令の演算を実行する算術論理・比較演算部 78 a ~ 78 c と、算術論理・比較演算部 78 a ~ 78 c と同様、S I M D 型命令の実行が可能であり、ビット精度を落とさないように、最長で 65 ビットで累算する乗算・積和演算部 80 a および 80 b とを含む。

【0038】

実行部 70 は、さらに、各々、データの算術シフト（2 の補数体系のシフト）または論理シフト（符号なしシフト）を実行するバレルシフタ 82 a ～ 82 b と、除算器 84 と、データメモリ 100 に接続され、データメモリ 100 との間でデータの受渡しを行うオペランドアクセス部 88 と、32 ビット幅のデータバス 90（L1 バス、R1 バス、L2 バス、R2 バス、L3 バス、R3 バス）と、32 ビット幅のデータバス 92（D1 バス、D2 バス、D3 バス）とを含む。

【0039】

レジスタファイル 76 は、32 個の 32 ビットレジスタ R0 ～ R31 を含む。L1 バス、R1 バス、L2 バス、R2 バス、L3 バスおよび R3 バスにデータを出力するレジスタファイル 76 内のレジスタの選択は、実行制御部 72 よりレジスタファイル 76 に供給される制御信号 CL1、CR1、CL2、CR2、CL3 および CR3 によりそれぞれ行なわれる。また、D1 バス、D2 バスおよび D3 バスを流れるデータが書き込まれるレジスタの選択は、実行制御部 72 よりレジスタファイル 76 に供給される制御信号 CD1、CD2 および CD3 によりそれぞれ行なわれる。

【0040】

算術論理・比較演算部 78 a の 2 つの入力ポートは、L1 バスおよび R1 バスにそれぞれ接続され、その出力ポートは、D1 バスに接続されている。算術論理・比較演算部 78 b の 2 つの入力ポートは、L2 バスおよび R2 バスにそれぞれ接続され、その出力ポートは、D2 バスに接続されている。算術論理・比較演算部 78 c の 2 つの入力ポートは、L3 バスおよび R3 バスにそれぞれ接続され、その出力ポートは、D3 バスに接続されている。

【0041】

乗算・積和演算部 80 a の 4 つの入力ポートは、L1 バス、R1 バス、L2 バスおよび R2 バスにそれぞれ接続され、その 2 つの出力ポートは、D1 バスおよび D2 バスにそれぞれ接続されている。乗算・積和演算部 80 b の 4 つの入力ポートは、L2 バス、R2 バス、L3 バスおよび R3 バスにそれぞれ接続され、その 2 つの出力ポートは、D2 バスおよび D3 バスにそれぞれ接続されている。

【0042】

バレルシフタ 82 a の 2 つの入力ポートは、L 1 バスおよび R 1 バスにそれぞれ接続され、その出力ポートは、D 1 バスに接続されている。バレルシフタ 82 b の 2 つの入力ポートは、L 2 バスおよび R 2 バスにそれぞれ接続され、その出力ポートは、D 2 バスに接続されている。バレルシフタ 82 c の 2 つの入力ポートは、L 3 バスおよび R 3 バスにそれぞれ接続され、その出力ポートは、D 3 バスに接続されている。

【0043】

除算器 84 の 2 つの入力ポートは、L 1 バスおよび R 1 バスにそれぞれ接続され、その出力ポートは、D 1 バスに接続されている。

オペランドアクセス部 88 およびデータメモリ 100 は、OA (Operand Address) バス 96 および OD (Operand Data) バス 94 により接続されている。OA バス 96 および OD バス 94 はそれぞれ 32 ビット幅である。また、オペランドアクセス部 88 は、OA バス 96 を介してデータメモリ 100 のアドレスを指定し、OD バス 94 を介して、当該アドレスのデータの読み書きを行なう。

【0044】

また、オペランドアクセス部 88 は、D 1 バス、D 2 バス、D 3 バス、L 1 バスおよび R 1 バスに接続され、いずれかのバスとの間でデータの受渡しを行なう。

【0045】

プロセッサ 30 は、3 命令を並列実行可能であるが、後述するように、パイプライン動作のうち、並列実行可能な命令割り当てステージ、デコードステージ、実行ステージおよび書き込みステージからなる 1 組のパイプライン処理を実行させる回路の集合を本明細書中で「スロット」と定義する。したがって、プロセッサ 30 は、第 1 ～ 第 3 スロットの 3 つのスロットを有する。命令レジスタ 56 a および命令デコーダ 64 a の組は第 1 スロット、命令レジスタ 56 b および命令デコーダ 64 b の組は第 2 スロット、命令レジスタ 56 c および命令デコーダ 64 c の組は第 3 スロットにそれぞれ属するものとする。

【0046】

各スロットには、デフォルト論理と呼ばれる命令が割り当てられており、同一のスロットでは極力同一の命令が実行されるように、命令スケジューリングが行なわれる。例えば、第1スロットにはメモリアクセスに関する命令（デフォルト論理）、第2スロットには乗算に関するデフォルト論理、第3スロットにはその他のデフォルト論理が割り当てられている。なお、デフォルト論理は、図1を参照して説明したオペレーション群と一対一に対応している。すなわち、先頭2ビットが“01”、“10”および“11”の命令は、それぞれ第1スロット、第2スロットおよび第3スロットのデフォルト論理である。

【0047】

第1スロットのデフォルト論理としては、`ld`（ロード命令）、`st`（ストア命令）などがある。第2スロットのデフォルト論理としては、`mul1`、`mul2`（乗算命令）などがある。第3スロットのデフォルト論理としては、`add1`、`add2`（加算命令）、`sub1`、`sub2`（減算命令）、`mov1`、`mov2`（レジスタ間の転送命令）などがある。

【0048】

図4は、パケットに含まれる並列実行境界情報について説明するための図である。命令メモリ40には、パケット112および114の順でパケットが記憶されているものとする。このうち、パケット112の命令2およびパケット114の命令5の並列実行境界情報は“1”であるものとし、それ以外の命令の並列実行境界情報は“0”であるものとする。

【0049】

命令フェッチ部52は、PC部74のプログラムカウンタの値に基づいて、パケット112、パケット114の順でパケットを読み込み、順次、命令バッファ54に供給する。実行部70では並列実行境界情報が“1”までの命令が並列実行される。

【0050】

図5は、パケットの並列実行境界情報に基づいて作成された並列実行される命令の実行単位の一例を示す図である。図4および図5を参照して、パケット112および114を並列実行境界情報が“1”の命令の部分で区切ると、実行単位

122～126が生成される。したがって、命令バッファ54から命令レジスタ部56には、実行単位122～126の順で命令が供給されることとなる。これらの、命令の供給に関する制御は、命令発行制御部62が行なう。

【0051】

命令デコーダ64a～64cは、命令レジスタ56a～56cに保持された命令のオペコードをそれぞれ解読し、制御信号を実行制御部72に出力する。実行制御部72は、命令デコーダ64a～64cでの解析結果に基づいて、実行部70の構成要素の各種制御を行なう。

【0052】

例えば、命令“addl R3, R0”について考える。この命令の意味は、レジスタR3の値とレジスタR0の値とを加算し、結果をレジスタR0に書き込むというものであるが、この場合、実行制御部72は、以下のような制御を一例として行なう。実行制御部72は、レジスタR3に保持された値をL1バスに出力するための制御信号CL1をレジスタファイル76に供給する。また、実行制御部72は、レジスタR0に保持された値をR1バスに出力するための制御信号CR1をレジスタファイル76に供給する。

【0053】

さらに、実行制御部72は、D1バスを介して得られる実行結果をレジスタR0に書き込むための制御信号CD1をレジスタファイル76に供給する。さらにまた、実行制御部72は算術論理・比較演算部78aを制御し、L1バスおよびL2バスを介してレジスタR3およびR0の値を受け取り、加算した後、加算結果をD1バスを介してレジスタR0に書き込む。

【0054】

図6は、算術論理・比較演算部78a～78cの概略構成を示すブロック図である。図6および図2を参照して、算術論理・比較演算部78a～78cの各々は、データバス90を介してレジスタファイル76に接続されたALU (Arithmetic and Logical Unit) 部132と、ALU部132およびデータバス92を介してレジスタファイル76に接続され、飽和、最大・最小値検出、絶対値生成処理を行なう飽和处理部134と、ALU部132に接続され、オーバーフロー

の検出とコンディションフラグの生成を行なうフラグ部136とを含む。

【0055】

図7は、バレルシフタ82a～82cの概略構成を示すブロック図である。図7および図2を参照して、バレルシフタ82a～82cの各々は、32ビットのデータを保持するアキュムレータM0およびM1を有するアキュムレータ部142と、アキュムレータM0およびデータバス90を介してレジスタファイル76に接続され、アキュムレータM0またはレジスタの値を受けるセクタ146と、アキュムレータM1およびデータバス90を介してレジスタファイル76に接続され、アキュムレータM1およびレジスタの値を受けるセクタ148と、セクタ146の出力に接続された上位バレルシフタ150と、セクタ148の出力に接続された下位バレルシフタ152と、上位バレルシフタ150および下位バレルシフタ152の出力に接続された飽和処理部154とを含む。

【0056】

飽和処理部154の出力は、アキュムレータ部142とデータバス92を介してレジスタファイル76とに接続されている。

バレルシフタ82a～82cの各々は、構成部品を動作させることにより、データの算術シフト(2の補数体系のシフト)または論理シフト(符号なしシフト)を実行する。通常は、32ビットもしくは、64ビットのデータを入出力としている。レジスタファイル76内のレジスタまたはアキュムレータ部142内のアキュムレータに格納された被シフトデータに対して、別のレジスタまたは即値でシフト量が指定される。データは、左63ビット～右63ビットの算術または論理シフトが行われ、入力ビット長で出力される。

【0057】

また、バレルシフタ82a～82cの各々は、SIMD型命令に対して、8、16、32、64ビットのデータをシフトすることができる。例えば、8ビットデータのシフトを4並列で処理することができる。

【0058】

算術シフトは、2の補数体系のシフトであり、加算や減算時の小数点の位置合わせや、2のべき乗の乗算(2、2の2乗、2の(−1)乗、2の(−2)乗倍

など)等のために行われる。

【0059】

図8は、除算器84の概略構成を示すブロック図である。図8および図2を参照して、除算器84は、32ビットのデータを保持するアキュムレータM0およびM1を有するアキュムレータ部162と、アキュムレータ部162ならびにデータバス90および92を介してレジスタファイル76に接続された除算部164とを含む。

【0060】

除算器84は、被除数を64ビット、除数を32ビットとし、商と剰余を32ビットずつ出力する。商と剰余を求めるまでに34サイクルを必要とする。符号付き、符号なし、両方のデータを扱うことが可能である。ただし、被除数と除数において符号の有無の設定は共通とする。その他、オーバーフローフラグ、0除算フラグを出力する機能を有する。

【0061】

図9は、乗算・積和演算部80aおよび80bの概略構成を示すブロック図である。図9および図2を参照して、乗算・積和演算部80aおよび80bの各々は、64ビットのデータを保持するアキュムレータM0およびM1を有するアキュムレータ部172と、各々データバス90を介してレジスタファイル76に接続された2入力の32ビット乗算器(MUL)174aおよび174bとを含む。

【0062】

乗算・積和演算部80aおよび80bの各々は、さらに、乗算器174aの出力およびアキュムレータ部172に接続された64ビット加算器(Adder)176aと、乗算器174bの出力およびアキュムレータ部172に接続された64ビット加算器176bと、64ビット加算器176aおよび64ビット加算器176bの出力に接続された64ビット加算器176cと、64ビット加算器176bおよび176cの出力に接続されたセクタ178と、加算器176aの出力、セクタ178の出力、アキュムレータ部172およびデータバス92を介してレジスタファイル76に接続された飽和处理部(Saturation)180とを含む。

【0063】

乗算・積和演算部 80 a および 80 b の各々は、以下の乗算、積和演算を行う。

。

- ・ 32 × 32 ビットの signed の乗算、積和、積差演算
- ・ 32 × 32 ビットの unsigned の乗算
- ・ 16 × 16 ビットの 2 並列の signed の乗算、積和、積差演算
- ・ 32 × 16 ビットの 2 並列の signed の乗算、積和、積差演算

これらの演算を整数、固定小数点フォーマットのデータに対して行う。また、これらの演算に対し、丸め、飽和を行う。

【0064】

図 10 は、このようなプロセッサ 30 による命令実行時の各パイプライン動作を示すタイミング図である。図 2 および図 10 を参照して、命令フェッチステージでは、命令フェッチ部 52 が、PC 部 74 に保持されたプログラムカウンタで指定されるアドレスの命令メモリ 40 をアクセスし、パケットを命令バッファ 54 に転送する。命令割り当てステージでは、命令バッファ 54 に保持された命令が命令レジスタ 56 a ~ 56 c に割り当てられる。デコードステージでは、命令レジスタ 56 a ~ 56 c に割り当てられた命令が、命令発行制御部 62 からの制御に従い、命令デコーダ 64 a ~ 64 c でそれぞれデコードされる。実行ステージでは、命令デコーダ 64 a ~ 64 c でのデコード結果に基づいて、実行制御部 72 が実行部 70 の構成部品を動作させ、各種演算を実行する。書き込みステージでは、演算結果をデータメモリ 100 またはレジスタファイル 76 に格納する。これらの処理により、最高 3 並列のパイプライン処理が実行可能になる。

【0065】

図 11 は、プロセッサ 30 で実行される命令、処理の内容およびそのビットパターンを示す図である。命令 “ld Rs, Rd” は、データメモリ 100 の、図 1 で示したオペレーションの Rs フィールドで指定されるレジスタ（以下「レジスタ Rs」という。以下同様である。）の値の番地のデータメモリ 100 に記憶されたデータを、レジスタ Rd にロードする処理を示す。そのビットパターンは、図 11 で図示したとおりである。

【0066】

なお、図11のビットパターンのうち、先頭2ビット（30および31ビット目）は、オペレーション群を指定するために用いられ、0ビット目は、並列実行境界情報を指定するために用いられる。上述の先頭2ビットが“01”のオペレーションは、メモリアクセスに関するものである。先頭2ビットが“10”のオペレーションは、乗算に関するものである。先頭2ビットが“11”のオペレーションは、その他の演算に関するものである。

【0067】

命令“s t R s, R d”は、レジスタR sの値を、データメモリ100のレジスタR dで指定される番地にストアする処理を示す。

命令“m u l 1 R s, R d”は、レジスタR sの値とレジスタR dの値との積をレジスタR dに書き込む処理を示す。命令“m u l 2 R s 1, R s 2, R d”は、レジスタR s 1の値とレジスタR s 2の値との積をレジスタR dに書き込む処理を示す。

【0068】

命令“a d d 1 R s, R d”は、レジスタR sの値とレジスタR dの値との和をレジスタR dに書き込む処理を示す。命令“a d d 2 R s 1, R s 2, R d”は、レジスタR s 1の値とレジスタR s 2の値との和をレジスタR dに書き込む処理を示す。

【0069】

命令“s u b 1 R s, R d”は、レジスタR sの値とレジスタR dの値との差をレジスタR dに書き込む処理を示す。命令“s u b 2 R s 1, R s 2, R d”は、レジスタR s 1の値とレジスタR s 2の値との差をレジスタR dに書き込む処理を示す。

【0070】

命令“m o v 1 R s, R d”は、レジスタR sの値をレジスタR dに書き込む処理を示す。命令“m o v 2 I m m, R d”は、I m mフィールドの値をレジスタR dに書き込む処理を示す。

【0071】

命令 “div Rs, Rd” は、レジスタ Rs の値をレジスタ Rd の値で除した商をレジスタ Rd に書き込む処理を示す。命令 “mod Rs, Rd” は、レジスタ Rs の値をレジスタ Rd の値で除した剰余をレジスタ Rd に書き込む処理を示す。

【0072】

[コンパイラ]

次に、上述のプロセッサ 30 をターゲットとする本実施の形態に係るコンパイラの一例について図 12 ～図 38 を用いて説明する。

【0073】

[コンパイラの全体構成]

図 12 は、本実施の形態に係るコンパイラ 200 の構成を示す機能ブロック図である。このコンパイラ 200 は、C/C++ 言語等の高級言語で記述されたソースプログラム 202 を、上述のプロセッサ 30 をターゲットプロセッサとする機械語プログラム 204 に変換するクロスコンパイラであり、パーソナルコンピュータ等のコンピュータ上で実行されるプログラムによって実現され、大きく分けて、パーサー部 210 と、中間コード変換部 220 と、最適化部 230 と、コード生成部 240 とから構成される。

【0074】

パーサー部 210 は、コンパイルの対象となるソースプログラム 202（インクルードされるヘッダファイルを含む）に対して、予約語（キーワード）等を抽出して字句解析する前置処理部であり、通常のコンパイラが備える解析機能を有する。

【0075】

中間コード変換部 220 は、パーサー部 210 に接続され、パーサー部 210 から渡されたソースプログラム 202 の各ステートメントを一定規則に基づいて中間コードに変換する処理部である。ここで、中間コードは、典型的には、関数呼び出しの形式で表現されるコード（例えば、「+(int a, int b)」を示すコード；「整数 a に整数 b を加算する」ことを示す。）である。

【0076】

最適化部 2 3 0 は、中間コード変換部 2 2 0 に接続され、中間コード変換部 2 2 0 から出力された中間コードについて、命令のオペコードに着目し、命令間の依存関係を崩すことなくプロセッサ 3 0 の消費電力が小さくなるように、命令の配置を行なう命令スケジューリング部 2 3 2 と、命令スケジューリング部 2 3 2 に接続され、命令スケジューリング部 2 3 2 でのスケジュール結果について、命令のレジスタフィールドに着目して、プロセッサ 3 0 の消費電力が小さくなるようにレジスタを割り付けるレジスタ割付部 2 3 4 とを含む。

【 0 0 7 7 】

最適化部 2 3 0 は、さらに、レジスタ割付部 2 3 4 に接続され、レジスタが割り付けられたスケジュール結果について、命令のビットパターンに着目し、命令の依存関係を崩すことなくプロセッサ 3 0 の消費電力が小さくなるように、命令の再配置を行なう命令再スケジューリング部 2 3 6 と、命令再スケジューリング部 2 3 6 に接続され、命令再スケジューリング部 2 3 6 のスケジュール結果について、一定サイクル以上停止しているスロットを検出し、その前後に、当該スロットを停止および復帰させる命令を挿入するスロット停止・復帰命令生成部 2 3 8 とを含む。

【 0 0 7 8 】

最適化部 2 3 0 は、さらに、スロット停止・復帰命令生成部 2 3 8 に接続され、スケジュール結果に基づいて、配置された命令の並列実行境界情報を設定する並列実行境界情報設定部 2 3 9 と、命令スケジューリング部 2 3 2、レジスタ割付部 2 3 4 および命令再スケジューリング部 2 3 6 に接続され、スケジュール結果を各サイクルごとに消費電力が小さくなるように配置しなおすサイクル内配置調整処理部 2 3 7 とを含む。

【 0 0 7 9 】

なお、後述する最適化部 2 3 0 での処理は、基本ブロック単位に行なわれる。基本ブロックとは、たとえば式や代入文の並びのような、途中から外部への分岐が起こらず、また、外部から途中への分岐も起こらないプログラムの単位を言う。

【 0 0 8 0 】

コード生成部 240 は、最適化部 230 の並列実行境界情報設定部 239 に接続され、並列実行境界情報設定部 239 から出力された中間コードに対して、内部に保持する変換テーブル等を参照することで、全ての中間コードを機械語命令に置き換えることで、機械語プログラム 204 を生成する。

【0081】

次に、以上のように構成されたコンパイラ 200 の特徴的な動作について、具体的な例を示しながら説明する。

【0082】

[命令スケジューリング部]

図 13 は、命令スケジューリング部 232 の動作を示すフローチャートである。命令スケジューリング部 232 の処理では、レジスタのスケジューリングは行なわず、レジスタの個数は無限にあると想定して処理が行なわれる。したがって、以下の説明では、命令スケジューリング部 232 でスケジューリングされるレジスタには V r (Virtual Register) 0、V r 1 など、先頭に V r が付されるものとする。

【0083】

命令スケジューリング部 232 は、中間コード変換部 220 で生成された中間コードに基づいて、命令の依存グラフを作成する（ステップ S 2（以下「ステップ」を省略する。））。依存グラフとは、命令間の依存関係を示したグラフであり、命令ごとにノードを割り付け、依存関係のある命令をエッジで結んだ有向グラフである。依存グラフに関しては、周知の技術である。したがって、その詳細な説明はここでは繰返さない。たとえば、ここでは、図 14（a）に示されるような 3 つの有向グラフからなる依存グラフが作成されるものとする。

【0084】

命令スケジューリング部 232 は、依存グラフの中から実行可能な命令（ノード）を選択し、そのうち、各スロットのデフォルト論理に合致するように 1 サイクル目の命令をスケジューリングする（S 4）。例えば、図 14（a）の依存グラフでは、ノード N 1、N 6、N 7、N 11 および N 12 の命令のノードがスケジューリング可能であるが、そのうち、ノード N 1 がメモリアクセスに関する命

令であり、ノードN11が乗算命令であり、ノードN6がシフト命令であるとする。この場合、ノードN1、N11およびN6がそれぞれ1サイクル目の第1～第3スロットにそれぞれ配置される。配置済みのノードにはフラグが付され、依存グラフは図14(b)のように更新される。1サイクル目の命令スケジューリング(S4)の後、図15に示されるような命令のスケジュール結果が得られる。

【0085】

命令スケジューリング部232は、依存グラフを参照し、配置候補命令集合を生成する(S8)。すなわち、図14(b)の例では、ノードN2、N7、N8およびN12で示される命令が配置候補命令集合となる。

【0086】

命令スケジューリング部232は、配置候補命令集合の中から後述するアルゴリズムに従い、最適な命令を1つ取り出す(S12)。

命令スケジューリング部232は、取り出された最適命令が実際に配置可能かを判断する(S14)。配置可能か否かの判断は、最適命令を配置したと仮定した場合の着目サイクルの命令数が、1つ前のサイクルに配置された命令数を超えないか否かにより判断される。これにより、同一数の命令が配置されたサイクルが連続することとなる。

【0087】

配置可能と判断した場合には(S14でYES)、その最適命令を仮配置し、配置候補命令集合から削除する(S16)。その後、命令スケジューリング部232は、さらに命令を配置することが可能か否かを、上述の判断処理(S14)と同様にして判断する(S18)。配置可能と判断した場合には(S18でYES)、依存グラフを参照し、新たな配置候補命令が生じた場合には、それを配置候補命令集合に追加する(S20)。以上の着目サイクルに対する命令仮配置処理を、配置候補命令がなくなるまで繰返す(S10～S22)。

【0088】

なお、最適命令の仮配置処理(S16)の後、これ以上、着目サイクルに命令を配置することができないと判断した場合には(S18でNO)、命令の仮配置

処理 (S10～S22) のループを抜ける。

【0089】

命令の仮配置処理 (S10～S22) の後、命令スケジューリング部232は、仮配置された命令を確定させ、配置候補命令集合に対するスケジューリングを終了する (S24)。その後、配置済みの命令に関しては、依存グラフの対応するノードに配置済みのフラグが付され、依存グラフの更新が行なわれる (S26)。

【0090】

命令スケジューリング部232は、一定サイクル以上、同一数の命令が連続配置されているか否かを判断する (S27)。一定サイクル以上、同一数の命令が連続配置されていると判断した場合 (たとえば、20サイクル以上2命令が連続配置されている場合や、10サイクル以上1命令が連続配置されている場合) には (S27でYES)、命令スケジューリング部232は、1サイクルに配置可能な命令の最大数 (以下「最大配置可能命令数」という。) を3に設定し (S28)、以降のサイクルでは、なるべく、1サイクルに3命令が配置されるようにする。以上の処理を、未配置命令がなくなるまで繰返す (S6～S29)。

【0091】

図16は、図13の最適命令取出し処理 (S12) の動作を示すフローチャートである。

命令スケジューリング部232は、配置候補命令の各々について、着目サイクルの1つ前のサイクルで配置された命令の各々との間で、オペコードのビットパターン間のハミング距離を求める (S42)。

【0092】

たとえば、図14 (b) を参照して、2サイクル目のスケジューリングの開始当初は、ノードN2、N7、N8およびN12が配置可能である。1サイクル目では、ノードN1、N6およびN11が配置されている。このため、ノードN1、N6およびN11とノードN2、N7、N8およびN12との間のすべての組み合わせについて、オペコードのビットパターン間のハミング距離が求められることになる。

【0093】

図17は、オペコードのビットパターン間のハミング距離の算出方法を説明するための図である。Nサイクル目にはすでに、命令“ld Vr11, Vr12”が配置済みであり、N+1サイクル目の配置候補命令は、“st Vr13, Vr14”および“addl Vr13, Vr14”であるとする。図17(a)を参照して、オペコード“ld”および“st”は、12、16、17、24および25ビット目のビットパターンが異なる。このため、ハミング距離は5である。同様にして、図17(b)を参照して、オペコード“ld”および“addl”は、16、17、18、20、25、26、28および31ビット目のビットパターンが異なる。このため、ハミング距離は8である。

【0094】

図18は、ビット長が異なるオペコード間でのハミング距離の算出方法を説明するための図である。Nサイクル目にはすでに、命令“ld Vr11, Vr12”が配置済みであり、N+1サイクル目の配置候補命令は、“mul2 Vr13, Vr14, Vr15”および“st Vr13, Vr14”であるとする。図18(a)を参照して、オペコード“ld”および“mul2”のように、ビット長が異なるオペコード間では、オペコードの重複部分のビットパターンについてハミング距離が計算される。したがって、オペコードの16ビット目から31ビット目までの値に基づいてハミング距離が算出される。オペコード“ld”および“mul2”は、16、18、19、22、23、25、26、27、28、30および31ビット目が異なる。このため、ハミング距離は11である。図18(b)を参照して、その他の配置候補命令“st Vr13, Vr14”についても、図18(a)の例との整合性を確保するため、オペコードの16ビット目から31ビット目までの値に基づいてハミング距離が算出される。オペコード“ld”および“st”は、16、17、24および25ビット目が異なる。このため、ハミング距離は4である。

【0095】

再度図16を参照して、命令スケジューリング部232は、最小ハミング距離を有する配置候補命令を特定する(S43)。図17および図18の例では、命

令 “s t V r 1 3, V r 1 4” がともに配置候補命令として特定される。

【0096】

命令スケジューリング部 232 は、最小ハミング距離を有する配置候補命令が 2 以上あるか否かを判断する (S44)。最小ハミング距離を有する配置候補命令が 1 つの場合には (S44 で NO)、その命令を最適命令とする (S56)。

【0097】

最小ハミング距離を有する配置候補命令が 2 つ以上ある場合には、(S44 で YES)、それらの配置候補命令のうち命令が配置されていない空きスロットのデフォルト論理に合致するものがあるか否かを判断する (S46)。

【0098】

デフォルト論理に合致する配置候補命令がなければ (S46 で NO)、最小ハミング距離を有する 2 以上の配置候補命令のいずれかを任意に選択し、最適命令とする (S54)。

【0099】

デフォルト論理に合致する配置候補命令があり、かつその個数が 1 つであれば (S46 で YES、S48 で NO)、デフォルト論理に合致する配置候補命令を最適命令とする (S52)。

【0100】

デフォルト論理に合致する配置候補命令があり、かつその個数が 2 つ以上であれば (S46 で YES、S48 で YES)、デフォルト論理に合致する 2 以上の配置可能命令のうちのいずれかを任意に選択して最適命令とする (S50)。

【0101】

[サイクル内配置調整処理部]

図 19 は、サイクル内配置調整処理部 237 の動作を示すフローチャートである。サイクル内配置調整処理部 237 は、命令スケジューリング部 232 でのスケジュール結果に基づいて、各サイクル内での命令の配置の調整を行なう。

【0102】

サイクル内配置調整処理部 237 は、スケジュール結果の 2 サイクル目から最終サイクルまでのうち、着目しているサイクルの 3 つの命令について並べ替えを

行ない、6通りの命令並びを作成する（S61）。図20は、このようにして作成された6通りの命令並びの一例を示す図である。

【0103】

サイクル内配置調整処理部237は、後述する6通りの命令並びの各々についてハミング距離の和を求める処理（S62～S67）を実行する。6通りの命令並びの各々について求められたハミング距離の和のうち最小のハミング距離の和をとる命令並びを選択し、その命令並びの並びになるように命令の並べ替えを行なう（S68）。以上の処理を、2サイクル目から最終サイクルまで繰返す（S60～S69）。

【0104】

次に、6通りの命令並びの各々についてハミング距離の和を求める処理（S62～S67）について説明する。サイクル内配置調整処理部237は、各命令並びの各スロットについて、着目命令と1つ前のサイクルの命令とのオペコードのビットパターン間のハミング距離を求める（S64）。ハミング距離を求める処理（S64）を3つのスロットの命令のすべてについて行ない（S63～S65）、3つのスロットの命令の各々についてハミング距離の和を求める（S66）。以上の処理を、6通りの命令並びのすべてについて行なう（S62～S67）。

【0105】

図21は、配置された命令の一例を示す図である。Nサイクル目には、第1スロット、第2スロットおよび第3スロットで実行される命令として、“ld Vr10, Vr11”、“subl Vr12, Vr13”および“addl Vr14, Vr15”がそれぞれ配置されているものとする。N+1サイクル目には、第1スロット、第2スロットおよび第3スロットで実行される命令として、“st Vr16, Vr17”、“mul Vr18, Vr19”および“mod Vr20, Vr21”がそれぞれ配置されているものとする。

【0106】

図22は、命令並び作成処理（S61）を説明するための図である。たとえば、図21に示すN+1サイクル目に配置された3つの命令より、図22（a）～

(f) に示す 6 つの命令並びが作成される。

【0107】

図 23 は、オペコードのハミング距離算出処理 (S64) を説明するための図である。たとえば、図 21 に示す N サイクル目の命令並びと、図 22 (c) に示す N+1 サイクル目の命令並びとの間で、スロットごとにオペコードのハミング距離を算出すると、第 1 スロット、第 2 スロットおよび第 3 スロットにおけるハミング距離は、それぞれ 10、9 および 5 となる。

【0108】

したがって、図 23 の例におけるハミング距離の和は 24 となる。ハミング距離和算出処理 (S66) では、このようにして、図 21 に示す N サイクル目の命令並びと、図 22 (a) ~ (f) に示す 6 通りの命令並びの各々との間でハミング距離の和が求められ、それぞれ、14、16、24、22、24 および 20 となる。命令並び選択処理 (S68) では、6 通りの命令並びのうち、最小のハミング距離の和をとる図 22 (a) の命令並びが選択される。

【0109】

[レジスタ割付部]

図 24 は、レジスタ割付部 234 の動作を示すフローチャートである。レジスタ割付部 234 では、命令スケジューリング部 232 およびサイクル内配置調整処理部 237 でのスケジュール結果に基づいて、実際にレジスタの割付を行っていく。

【0110】

レジスタ割付部 234 は、ソースプログラム 202 から割付対象 (変数) を抜き出し、各割付対象の生存区間およびその優先度を求める (S72)。生存区間とは、プログラム中で、変数が定義されてから、その参照が終了するまでの区間を言う。したがって、同一の変数であっても、複数の生存区間が存在する場合がある。優先度とは、割付対象の生存区間長およびその参照頻度で決定される。その詳細な説明は、本発明の本質的事項ではないため、省略する。

【0111】

レジスタ割付部 234 は、割付対象より干渉グラフを作成する (S74)。干

渉グラフとは、同一のレジスタを割り付けることができない割付対象の条件を示したグラフである。次に、干渉グラフの作成方法について説明する。

【0112】

図25は、割付対象となる変数の生存区間を示す図である。ここでは、変数I、JおよびKの3つの変数を割付対象とした例を示す。

変数Iは、ステップT1で定義されて、ステップT5で最終参照される。また、変数Iは、ステップT8で定義されて、ステップT10で最終参照される。したがって、変数Iは、2つの生存区間を有することとなる。先の生存区間での変数Iを変数I1と定義し、後の生存区間での変数を変数I2と定義することとする。変数Jは、ステップT2で定義されて、ステップT4で最終参照される。

【0113】

変数Kは、ステップT3で定義されて、ステップT6で最終参照される。また、変数Kは、ステップT7で定義されてステップT9で最終参照される。したがって、変数Iと同様、変数Kは2つの生存区間を有することとなる。先の生存区間での変数Kを変数K1と定義し、後の生存区間での変数Kを変数K2と定義する。

【0114】

変数I1、I2、J、K1およびK2には、以下に示すような生存区間の重なりが生じる。すなわち、変数I1およびJの生存区間は、ステップT2～T4で重なりを有する。変数JおよびK1の生存区間は、ステップT3～T4で重なりを有する。変数I1およびK1の生存区間は、ステップT3～T5で重なりを有する。変数I2およびK2は、ステップT8～T9で重なりを有する。このように、生存区間が重なる変数同士は、同一のレジスタに割り付けることはできない。このため、割付対象となる変数をノードとし、生存区間が重なる変数同士をエッジで結んだものが干渉グラフとなる。

【0115】

図26は、図25の例に基づいて作成された変数の干渉グラフを示す図である。ノードI1、K1およびJは相互にエッジにより接続されている。このため、変数I1、K1およびJの間には相互に生存区間が重なる区間があり、これら3

つの変数に同じレジスタを割り付けることはできないことがわかる。同様に、ノード I 2 および K 2 はエッジにより接続されている。このため、変数 I 2 および K 2 に同じレジスタを割り付けることはできないことがわかる。

【0116】

しかし、エッジにより接続されていないノード間には依存関係が存在しない。たとえば、ノード J および K 2 はエッジにより接続されていない。このため、変数 J および K 2 には生存区間の重なりがなく、同じレジスタを割り付けることができることがわかる。

【0117】

再度図 2 4 を参照して、レジスタ割付部 2 3 4 は、レジスタ割付を行っていない割付対象のうち、優先度が最も高い割付対象を選択する (S 8 0)。命令スケジューリング部 2 3 2 は、割付対象を割り付けるレジスタとして、同一スロットで割付対象を参照する命令の直前に実行される命令のうち、同一フィールドのレジスタ番号と同一番号のレジスタが割り付け可能か否かを判断する (S 8 2)。割り付け可能か否かの判断は、上述した干渉グラフを参照することにより行われる。

【0118】

図 2 7 は、命令スケジューリングの途中結果を示す図である。たとえば、図 2 7 (a) を参照して、現在の割付対象は、第 1 スロットの (N+1) 番目のサイクルのソースオペランド (レジスタ V r 5) に割り付けられるものとする。レジスタ V r 5 は、上述したように仮に設けられたレジスタである。このため、図 2 4 のレジスタ割付可能判断処理 (S 8 2) では、割付対象として、N 番目のサイクルの同一フィールドで使用されるレジスタ (ここでは、レジスタ R 0) が割り付け可能か否かを判断することになる。図 2 7 (b) は、V r 5 にレジスタ R 0 を割付けた場合の命令のビットパターンを示している。このように、連続するサイクル間で同一のレジスタをアクセスすると、レジスタの特性により消費電力を削減することができる。

【0119】

同一番号のレジスタが割り付け可能と判断された場合には (S 8 2 で Y E S)

、レジスタ割付部 234 は、割付対象に、上述の同一番号のレジスタを割り付ける (S84)。同一番号のレジスタを割り付けることができないと判断された場合には (S82 で NO)、レジスタ割付部 234 は、割付可能なレジスタのレジスタ番号 (2 進表現) の中で、先行サイクルの同スロットの同一フィールドのレジスタ番号との間のハミング距離が最小となるものを求める (S86)。図 27 (c) は、レジスタ R0 のレジスタ番号 (00000) とのハミング距離が最小となるレジスタ番号 (00001) を有するレジスタ R1 が、使用可能なレジスタの中から選択された例を示している。

【0120】

ハミング距離が最小となる割付可能なレジスタが 1 つしかない場合には (S88 で NO)、割付対象に当該レジスタを割り付ける (S92)。ハミング距離が最小となる割付可能なレジスタが 2 つ以上ある場合には (S88 で YES)、2 つ以上の割付可能なレジスタのいずれかを任意に選択し、割付対象に割り付ける (S90)。以上の処理を、割付対象がなくなるまで行なう (S78 ~ S94)。

【0121】

レジスタ割付部 234 での処理の後、サイクル内配置調整処理部 237 は、レジスタ割付部 234 でのスケジュール結果に基づいて、各サイクル内での命令の配置の調整を行なう。サイクル内配置調整処理部 237 で実行される処理は、図 19 および図 20 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【0122】

[命令再スケジューリング部]

図 28 は、命令再スケジューリング部 236 の動作を示すフローチャートである。命令再スケジューリング部 236 は、命令スケジューリング部 232、レジスタ割付部 234 およびサイクル内配置調整処理部 237 で実行された処理により、プロセッサ 30 で動作可能にスケジューリングされた命令のスケジュール結果を、再度スケジューリングしなおす処理を行なう。すなわち、命令再スケジューリング部 236 は、レジスタ割付部 234 にて実レジスタが確定した命令列に

対して、再度スケジューリングを行なうものである。

【0123】

命令再スケジューリング部236は、スケジュール結果の中から冗長な命令を削除する(S112)。たとえば、命令“movl R0, R0”は、レジスタR0の内容をレジスタR0に書き込む処理であるため、冗長な命令である。また、同一サイクルの第1スロットの命令が“mov2 4, R1”であり、第2スロットの命令が“mov2 5, R1”である場合には、それぞれ4および5をレジスタR1に書き込む命令である。本実施の形態では、番号の大きいほうのスロットの命令が優先的に実行されることとする。このため、第1スロットの命令“mov2 4, R1”は、冗長な命令である。

【0124】

冗長な命令を削除すると、命令の依存関係が変化する可能性がある。このため、命令再スケジューリング部236は、依存グラフの再構築を行なう(S114)。命令再スケジューリング部236は、依存グラフの中から実行可能な命令(ノード)を選択し、そのうち、各スロットのデフォルト論理に合致するように1サイクル目の命令をスケジューリングする(S115)。1サイクル目の命令に対応する依存グラフのノードには、配置済みのフラグが付される。

【0125】

命令再スケジューリング部236は、依存グラフを参照し、配置候補命令集合を生成する(S118)。命令再スケジューリング部236は、配置候補命令集合の中から後述するアルゴリズムに従い、最適な命令を1つ取り出す(S122)。

【0126】

命令再スケジューリング部236は、取り出された最適命令が実際に配置可能か否かを判断する(S124)。配置可能か否かの判断は、図13のS14の判断と同様である。このため、その詳細な説明はここでは繰返さない。

【0127】

配置可能と判断した場合には(S124でYES)、その最適命令を仮配置し、配置候補命令集合から削除する(S126)。その後、命令再スケジューリン

グ部 236 は、さらに命令を配置することが可能か否かを、上述の配置可能判断 (S124) と同様にして判断する (S128)。配置可能と判断した場合には (S128 で YES)、依存グラフを参照し、新たな配置候補命令が生じた場合には、それを配置候補命令集合に追加する (S130)。以上の処理を、配置候補命令がなくなるまで繰り返す (S120～S132)。

【0128】

なお、最適命令の仮配置処理 (S126) の後、これ以上、着目サイクルに命令を配置することができないと判断した場合には (S128 で NO)、最適命令の仮配置処理 (S120～S132) のループを抜ける。

【0129】

最適命令の仮配置処理 (S120～S132) の後、命令再スケジューリング部 236 は、仮配置された命令を確定させ、配置候補命令集合に対するスケジューリングを終了する (S134)。その後、配置済みの命令に関しては、依存グラフの対応するノードに配置済みのフラグが付され、依存グラフの更新が行なわれる (S136)。

【0130】

命令再スケジューリング部 236 は、一定サイクル以上、同一数の命令が連続配置されているか否かを判断する (S137)。一定サイクル以上、同一数の命令が連続配置されていると判断した場合には (S137 で YES)、命令再スケジューリング部 236 は、最大配置可能命令数を 3 に設定し (S138)、以降のサイクルでは、なるべく、1 サイクルに 3 命令が配置されるようにする。以上の処理を、未配置命令がなくなるまで繰り返す (S116～S139)。

図 29 は、図 28 の最適命令取出し処理 (S122) の動作を示すフローチャートである。命令再スケジューリング部 236 は、配置候補命令のうち、着目サイクルの 1 つ前のサイクルの同一スロットで実行される命令と比較して、同一のレジスタ番号を有するフィールドの個数を求め、当該個数が最大の配置候補命令を特定する (S152)。

【0131】

図 30 は、配置候補命令特定処理 (S152) を説明するための図である。N

サイクル目の第1スロットで実行される命令として“a d d 1 R 0, R 2”が配置されているものとし、N+1サイクル目の第1スロットに配置可能な命令として、図30(a)に示す“s u b 1 R 0, R 1”と、図30(b)に示す“d i v R 0, R 2”とがあるものとする。図30(a)に示すように、当該配置位置に命令“s u b 1 R 0, R 1”を配置した場合には、同一のレジスタ番号を有するフィールドは、レジスタR0（レジスタ番号00000）が配置されたフィールドのみである。このため、同一のレジスタ番号を有するフィールドの個数は1つである。図30(b)に示すように、当該配置位置に命令“d i v R 0, R 2”を配置した場合には、レジスタR0（レジスタ番号00000）およびレジスタR2（00010）がそれぞれ配置された2つのフィールドが同一のレジスタ番号を有する。このため、同一のレジスタ番号を有するフィールドの個数は2つである。

【0132】

当該個数が最大の配置候補命令が1つしかない場合には（S154でNO）、その配置候補命令を最適命令とする（S174）。

当該個数が最大の配置候補命令がないか、または2つ以上ある場合には（S154でYES）、命令再スケジューリング部236は、配置候補命令の各々について、1つ前のサイクルの同一スロットで実行される命令と比較して、命令のビットパターンのハミング距離が最小のものを求める（S156）。

【0133】

図31は、配置候補命令特定処理（S156）を説明するための図である。Nサイクル目の第1スロットで実行される命令として“m u l 1 R 3, R 10”が配置されているものとし、N+1サイクル目の第1スロットに配置可能な命令として、図31(a)に示す“a d d 1 R 2, R 4”と、図30(b)に示す“s u b 2 R 11, R 0, R 2”とがあるものとする。それぞれの命令のビットパターンは図示するとおりである。図31(a)に示すように、当該配置位置に命令“a d d 1 R 2, R 4”を配置した場合には、命令“m u l 1 R 3, R 10”とのハミング距離は10である。図31(b)に示すように、当該配置位置に命令“s u b 2 R 11, R 0, R 2”を配置した場合には、命令“m u

11 R3, R10”とのハミング距離は8である。このため、配置候補命令として“sub2 R11, R0, R2”が特定される。

【0134】

最小ハミング距離を有する配置候補命令が1つの場合には（S158でNO）、当該配置候補命令を最適命令とする（S172）。

最小ハミング距離を有する配置候補命令が2つ以上ある場合には（S158でYES）、2つ以上の配置候補命令のうち、当該配置候補命令が実行されるスロットのデフォルト論理に合致する配置候補命令を特定する（S160）。

【0135】

図32は、配置候補命令特定処理（S160）を説明するための図である。Nサイクル目の第1スロットで実行される命令として“st R1, R13”が配置されているものとし、N+1サイクル目の第1スロットに配置可能な命令として、図32（a）に示す“ld R30, R18”と、図32（b）に示す“sub1 R8, R2”とがあるものとする。それぞれの命令のビットパターンは図示するとおりである。第1スロットのデフォルト論理は、上述したようにメモリアクセスに関する命令である。これは、命令の先頭2ビットが“01”であることより判別可能である。命令“ld R30, R18”の先頭2ビットは“01”であるため、第1スロットのデフォルト論理に合致するが、命令“sub1 R8, R2”の先頭2ビットは“11”であるため、第1スロットのデフォルト論理には合致しない。このため、配置候補命令として“ld R30, R18”が特定される。

【0136】

デフォルト論理に合致する配置候補命令がなければ（S162でNO）、最小ハミング距離を有する配置候補命令のうちのいずれかを任意に選択し、最適命令とする（S170）。

【0137】

デフォルト論理に合致する配置候補命令があり、かつその個数が1つであれば（S162でYES、S164でNO）、デフォルト論理に合致する配置候補命令を最適命令とする（S168）。

【0138】

デフォルト論理に合致する配置候補命令があり、かつその個数が2つ以上であれば（S162でYES、S164でYES）、デフォルト論理に合致する配置候補命令のうちのいずれかを任意に選択し、最適命令とする（S166）。

【0139】

命令再スケジューリング部236での処理の後、サイクル内配置調整処理部237は、命令再スケジューリング部236でのスケジュール結果に基づいて、各サイクル内での命令の配置の調整を行なう。サイクル内配置調整処理部237で実行される処理は、図19および図20を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【0140】

以上、命令再スケジューリング部236の動作について説明を行なったが、コンパイル時のオプションまたはソースプログラム中に記述されたプラグマに従って、1つのサイクルで使用するスロットの個数の制限を行なってもよい。プラグマとは、プログラムの意味を変更することなくコンパイラへの最適化の指針を与える記述のことを言う。たとえば、例1に示すように、C言語で記述されたソースプログラムをコンパイルする際のオプションとして“-para”を設け、その後に続く数字でスロットの数を規定する。例1では、ソースプログラム“foo.c”がCコンパイラによりコンパイルされるが、スケジュール結果の各サイクルには、必ず2命令が配置されることとなる。

【0141】

また、例2に示すように、ソースプログラム中に記述された各関数について、使用されるスロットの個数をプラグマで定義してもよい。例2では、関数funcを実行する際に使用されるスロットの個数が1つと規定されている。このため、スケジュール結果のうち、関数funcを実行するサイクルの各々には、必ず1命令のみが配置されることとなる。

（例1）

```
cc -para 2 foo.c
```

（例2）

```
#pragma para=1 func
int func(void) {
    . . . . .
}
```

なお、オプションとプラグマとが同時に設定された場合には、値の小さいほうが優先されるようにしてもよい。たとえば、例1に示すソースプログラム“foo.c”中に、例2に示す関数funcおよびそのプラグマが指定されている場合には、原則として、2スロットの並列処理が実行されるが、関数funcを実行するサイクルでは1スロットのみで処理が実行されるように、スケジュール結果が作成される。

【0142】

また、オプションおよびプラグマに関しては、命令再スケジューリング部236のみならず、命令スケジューリング部232またはレジスタ割付部234での動作で考慮されるようにしてもよい。

【0143】

[スロット停止・復帰命令生成部]

図33は、スロット停止・復帰命令生成部238の動作を示すフローチャートである。スロット停止・復帰命令生成部238は、命令再スケジューリング部236でのスケジュール結果から一定サイクル（たとえば4サイクル）以上、特定の1つのスロットのみが連続使用されている区間を検出する（S182）。スロット停止・復帰命令生成部238は、上記区間の1サイクル前の空きスロット位置に残りの2つのスロットを停止させる命令を挿入する（S184）。1サイクル前に命令を挿入する空きスロット位置がない場合には、1サイクル追加し、上記命令を挿入する。

【0144】

次に、スロット停止・復帰命令生成部238は、上記区間の1サイクル後の空きスロット位置に停止させておいた2つのスロットを復帰させる命令を挿入する（S186）。サイクル後に命令を挿入する空きスロット位置がない場合には、1サイクル追加し、上記命令を追加する。

【0145】

図34は、命令が配置されたスケジュール結果の一例を示す図である。10サイクル目から18サイクル目までの9サイクルは第1スロットのみが連続使用されている。このため、9サイクル目の空きスロットに、第1スロットのみを動作させ残りの2つのスロットを停止させる命令（“set1 1”）が書き込まれる。また、19サイクル目の空きスロットに、残りの2つのスロットを復帰させる命令（“clear1 1”）が書き込まれる。図35は、図33の特定の1スロットのみが連続使用されている場合の処理（S182～S186）で命令が書き込まれたスケジュール結果の一例を示す図である。

【0146】

再度図33を参照して、スロット停止・復帰命令生成部238は、スケジュール結果から一定サイクル（たとえば4サイクル）以上、特定の2つのスロットのみが連続使用されている区間を検出する（S188）。スロット停止・復帰命令生成部238は、上記区間の1サイクル前の空きスロット位置に残りの1つのスロットを停止させる命令を挿入する（S190）。1サイクル前に命令を挿入する空きスロット位置がない場合には、1サイクル追加し、上記命令を挿入する。

【0147】

次に、スロット停止・復帰命令生成部238は、上記区間の1サイクル後の空きスロット位置に停止させておいた1つのスロットを復帰させる命令を挿入する（S192）。サイクル後に命令を挿入する空きスロット位置がない場合には、1サイクル追加し、上記命令を追加する。

【0148】

図35のスケジュール結果では、4サイクル目から8サイクル目までの5サイクルは第1および第2スロットのみが使用され、第3スロットは使用されていない。このため、その前後のサイクルに第3スロットを停止させる命令（“set2 12”）および復帰させる命令（“clear2 12”）をそれぞれ書き込む必要がある。しかし、3サイクル目および9サイクル目の双方ともにすべてのスロットに命令が配置されている。このため、スロット停止・復帰命令生成部238は、4サイクル目の前および8サイクル目の後に1サイクルずつ新たなサ

イクルを挿入し、上記 2 命令をそれぞれのサイクルに書き込む。図 36 は、図 33 の特定の 2 スロットのみが連続使用されている場合の処理（S188～S192）で命令が書き込まれたスケジュール結果の一例を示す図である。

【0149】

なお、本実施の形態では、命令は、第 1 スロット、第 2 スロット、第 3 スロットの順に配置されることを前提としている。このため、2 つのスロットが動作している場合には、必ず第 3 スロットが動作しておらず、1 つのスロットのみが動作している場合には、必ず第 2 スロットと第 3 スロットとが動作していないことになる。

【0150】

また、プロセッサ 30 には、32 ビットのプログラム状態レジスタ（図示せず）が設けられている。図 37 は、プログラム状態レジスタの一例を示す図である。たとえば、15 および 16 ビットの 2 ビットで動作しているスロットの数を表すことができる。この場合、図 37（a）～（d）は、動作しているスロットの数がそれぞれ 0～3であることを示している。

【0151】

図 38 はプログラム状態レジスタの他の一例を示す図である。このプログラム状態レジスタでは、14 ビット目が第 1 スロットに、15 ビット目が第 2 スロットに、16 ビット目が第 3 スロットに対応している。各ビットの値が「1」であれば、そのスロットが動作していることを示し、「0」であれば、そのスロットが停止していることを示す。たとえば、図 38（b）のプログラム状態レジスタでは、第 1 スロットが停止しており、第 2 および第 3 スロットが動作していることを示している。

【0152】

上述した命令“set 1”または“set 2”でプログラム状態レジスタに保持された値が書き換えられる。

以上、本実施の形態におけるコンパイラについて説明したが、コンパイラ 200 の各部は以下のように変形可能である。次に、その変形例について順次説明を行なう。

【0153】

[コンパイラの各部の変形例]

[命令再スケジューリング部 236 の動作の変形例]

本実施の形態では、図 28 および図 29 を参照して、命令再スケジューリング部 236 の動作について説明したが、図 29 を参照して説明した図 28 の最適命令取出し処理 (S122) の代わりに、図 39 に示す最適命令取出し処理を行なってもよい。

【0154】

図 39 は、図 28 の最適命令取出し処理 (S122) の他の動作を示すフローチャートである。

命令再スケジューリング部 236 は、図 29 の最小ハミング距離を求める処理 (S156) の代わりに、以下に示す方法で最小ハミング距離を求める。すなわち、命令再スケジューリング部 236 は、配置候補命令のうち、1 つ前のサイクルの同一スロットで実行される命令と比較して、レジスタフィールドのビットパターンのハミング距離が最小のものを求める (S212)。

【0155】

図 40 は、配置候補命令特定処理 (S212) を説明するための図である。N サイクル目の第 1 スロットで実行される命令として “a d d 1 R0, R2” が配置されているものとし、N+1 サイクル目の第 1 スロットに配置可能な命令として図 40 (a) に示す “s u b 1 R3, R1” と、図 40 (b) に示す “d i v R7, R1” とがあるものとする。それぞれの命令のビットパターンは図示するとおりである。図 40 (a) に示すように、当該配置位置に命令 “s u b 1 R3, R1” を配置した場合には、命令 “a d d 1 R0, R2” とのレジスタフィールド間のハミング距離は 4 である。図 40 (b) に示すように、当該配置位置に命令 “d i v R7, R1” を配置した場合には、命令 “a d d 1 R0, R2” とのレジスタフィールド間のハミング距離は 5 である。このため、配置候補命令として、“a d d 1 R0, R2” が特定される。

【0156】

その他の処理 (S152～S154、S158～S174) は、図 29 で説明

したものと同様である。このため、その詳細な説明はここでは繰返さない。

【0157】

[サイクル内配置調整処理部237の第1変形例]

サイクル内配置調整処理部237は、図19を参照して説明した処理の代わりに、図41に示す処理を実行してもよい。

【0158】

図41は、サイクル内配置調整処理部237の動作の第1変形例を示すフローチャートである。

サイクル内配置調整処理部237は、図19のハミング距離を求める処理（S64）の代わりに、以下に示す方法で最小ハミング距離を求める。すなわち、サイクル内配置調整処理部237は、各命令並びの各スロットについて、着目命令と1つ前のサイクルの命令との間で、ビットパターン間のハミング距離を求める（S222）。その他の処理（S60～S63、S65～S69）は、図19を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【0159】

図42は、命令のハミング距離算出処理（S222）を説明するための図である。たとえば、図21に示すNサイクル目の命令並びと、図22（c）に示すN+1サイクル目の命令並びとの間で、スロットごとに命令のハミング距離を算出すると、第1スロット、第2スロットおよび第3スロットにおけるハミング距離は、それぞれ12、11および11となる。

【0160】

したがって、図42の例におけるハミング距離の和は34である。ハミング距離和算出処理（S66）では、このようにして、図21に示すNサイクル目の命令並びと、図22（a）～（f）に示す6通りの命令並びの各々との間でハミング距離の和が求められ、それぞれ28、26、34、28、34および30となる。命令並び選択処理（S68）では、6通りの命令並びのうち、最小のハミング距離の和をとる図22（b）の命令並びが選択される。

【0161】

なお、本変形例のハミング距離を求める処理（S 2 2 2）では、レジスタが割付けられていることが前提となっている。このため、本変形例のサイクル内配置調整処理部 2 3 7 での処理は、レジスタが割付けられていない命令スケジューリング部 2 3 2 での処理の後に実行することはできず、レジスタ割付部 2 3 4 での処理の後または命令再スケジューリング部 2 3 6 での処理の後に実行される。

【0 1 6 2】

[サイクル内配置調整処理部 2 3 7 の第 2 変形例]

サイクル内配置調整処理部 2 3 7 は、図 1 9 を参照して説明した処理の代わりに、図 4 3 に示す処理を実行してもよい。

【0 1 6 3】

図 4 3 は、サイクル内配置調整処理部 2 3 7 の動作の第 2 変形例を示すフローチャートである。

サイクル内配置調整処理部 2 3 7 は、図 1 9 のハミング距離を求める処理（S 6 4）の代わりに、以下に示す方法で最小ハミング距離を求める。すなわち、サイクル内配置調整処理部 2 3 7 は、各命令並びの各スロットについて、着目命令と 1 つ前のサイクルの命令とのレジスタフィールドのビットパターン間のハミング距離を求める（S 2 3 2）。その他の処理（S 6 0～S 6 3、S 6 5～S 6 9）は、図 1 9 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【0 1 6 4】

図 4 4 は、レジスタフィールドのハミング距離算出処理（S 2 3 2）を説明するための図である。たとえば、図 2 1 に示す N サイクル目の命令並びと、図 2 2（c）に示す N+1 サイクル目の命令並びとの間で、スロットごとにレジスタフィールドのハミング距離を算出すると、第 1 スロット、第 2 スロットおよび第 3 スロットにおけるハミング距離は、それぞれ 2、2 および 6 となる。

【0 1 6 5】

したがって、図 4 4 の例におけるハミング距離の和は 10 となる。ハミング距離和算出処理（S 6 6）では、このようにして、図 2 1 に示す N サイクル目の命令並びと、図 2 2（a）～（f）に示す 6 通りの命令並びの各々との間でハミン

グ距離の和が求められ、それぞれ14、10、10、6、10および10となる。命令並び選択処理（S68）では、6通りの命令並びのうち、最小のハミング距離の和をとる図22（d）の命令並びが選択される。

【0166】

なお、本変形例のハミング距離を求める処理（S232）では、レジスタが割付けられていることが前提となっている。このため、本変形例のサイクル内配置調整処理部237での処理は、レジスタが割付けられていない命令スケジューリング部232での処理の後に実行することはできず、レジスタ割付部234での処理の後または命令再スケジューリング部236での処理の後に実行される。

【0167】

[サイクル内配置調整処理部237の第3変形例]

サイクル内配置調整処理部237は、図19を参照して説明した処理の代わりに、図45に示す処理を実行してもよい。

【0168】

図45は、サイクル内配置調整処理部237の動作の第3変形例を示すフローチャートである。

サイクル内配置調整処理部237は、図19のハミング距離を求める処理（S64）の代わりに、以下の処理を実行する。すなわち、サイクル内配置調整処理部237は、各命令並びの各スロットについて、着目命令と1つ前のサイクルの命令との間で、同一のレジスタ番号を有するレジスタフィールドの個数を求める（S242）。

【0169】

また、サイクル内配置調整処理部237は、図19のハミング距離の和を求める処理（S66）の代わりに、以下の処理を実行する。すなわち、サイクル内配置調整処理部237は、3つのスロットの命令の各々について求められた同一のレジスタ番号を有するレジスタフィールドの個数の和を求める（S244）。

【0170】

さらに、サイクル内配置調整処理部237は、図19の命令の並べ替え処理（S68）の代わりに、以下の処理を実行する。すなわち、サイクル内配置調整処

理部 237 は、6 通りの命令並びの各々について求められたレジスタフィールドの個数の和のうち、最大のレジスタフィールドの個数の和をとる命令並びを選択し、その命令並びの並びになるように命令の並べ替えを行なう (S246)。その他の処理 (S60～S63、S65、S67 および S69) は、図 19 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【0171】

図 46 は、配置された命令の一例を示す図である。N サイクル目には、第 1 スロット、第 2 スロットおよび第 3 スロットで実行される命令として、“ld R0, R1”、“subl R2, R3” および “addl R4, R5” がそれぞれ配置されているものとする。N+1 サイクル目には、第 1 スロット、第 2 スロットおよび第 3 スロットで実行される命令として、“st R5, R8”、“mul R2, R3” および “mod R0, R10” がそれぞれ配置されているものとする。

【0172】

図 47 は、命令並び作成処理 (S61) を説明するための図である。たとえば、図 46 に示す N+1 サイクル目に配置された 3 つの命令より、図 47 (a) ～ (f) に示す 6 つの命令並びが作成される。

【0173】

図 48 は、レジスタフィールド個数算出処理 (S242) を説明するための図である。たとえば、図 46 に示す N サイクル目の命令並びと、図 47 (f) に示す N+1 サイクル目の命令並びとの間で、スロットごとに同一のレジスタ番号を有するレジスタフィールドの個数を求める。第 1 スロットについては、レジスタ R0 が両サイクルの同一レジスタフィールドで共通し、その他レジスタフィールドのレジスタは異なるため、当該個数は 1 である。第 2 スロットについては、レジスタ R2 および R3 が両サイクルの同一レジスタフィールドで共通するため、当該個数は 2 である。第 3 スロットについては、同一レジスタフィールドで共通するレジスタがないため、当該個数は 0 である。

【0174】

したがって、図 4 8 の例における同一のレジスタ番号を有するレジスタフィールドの個数の和は 3 である。レジスタフィールド個数和算出処理 (S 2 4 4) では、このようにして、図 4 6 に示す N サイクル目の命令並びと、図 4 7 (a) ~ (f) に示す 6 通りの命令並びとの各々との間でレジスタフィールドの個数の和が求められ、それぞれ 0, 2, 0, 0, 0, 1 および 3 となる。命令並び選択処理 (S 2 4 6) では、6 通りの命令並びのうち、最大のレジスタフィールドの個数和をとる図 4 7 (f) に示す命令並びが選択される。

【0175】

なお、本変形例のレジスタフィールドの個数を求める処理 (S 2 4 2) は、レジスタが割付けられていることが前提となっている。このため、本変形例のサイクル内配置調整処理部 2 3 7 での処理は、レジスタが割付けられていない命令スケジューリング部 2 3 2 での処理の後に実行することはできず、レジスタ割付部 2 3 4 での処理の後または命令再スケジューリング部 2 3 6 での処理の後に実行される。

【0176】

[サイクル内配置調整処理部 2 3 7 の第 4 変形例]

サイクル内配置調整処理部 2 3 7 は、図 1 9 を参照して説明した処理の代わりに、図 4 9 に示す処理を実行してもよい。

【0177】

図 4 9 は、サイクル内配置調整処理部 2 3 7 の動作の第 4 変形例を示すフローチャートである。

サイクル内配置調整処理部 2 3 7 は、図 1 9 の命令並びごとにハミング距離の和を求める処理 (S 6 3 ~ S 6 6) の代わりに、以下の処理を実行する。すなわち、サイクル内配置調整処理部 2 3 7 は、着目している命令並びに含まれる命令のうち、スロットのデフォルト論理に合致する命令の個数を求める (S 2 5 2)。

【0178】

また、サイクル内配置調整処理部 2 3 7 は、図 1 9 の命令の並べ替え処理 (S 6 8) の代わりに、以下の処理を実行する。すなわち、サイクル内配置調整処理

部 237 は、6 通りの命令並びの各々について求められたデフォルト論理に合致する命令の個数のうち、最大個数をとる命令並びを選択し、その命令並びの並びになるように命令の並べ替えを行なう（S 254）。その他の処理（S 60～S 62、S 67 および S 69）は、図 19 を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【0179】

たとえば、命令並び作成処理（S 61）で図 47（a）～（f）に示す 6 つの命令並びが作成されたものとする。上述のとおり、命令並びに含まれる各命令が配置されたスロットのデフォルト論理に合致するか否かは、命令の先頭 2 ビットを参照することにより判別可能である。たとえば、図 47（b）に示す命令並びでは、第 1 スロットに配置された命令の先頭 2 ビットが“01”であるため、当該スロットのデフォルト論理に合致するが、第 2 スロットおよび第 3 スロットに配置された命令の先頭 2 ビットはそれぞれ“11”および“10”であるため、当該スロットのデフォルト論理には合致しない。このため、当該スロットでデフォルト論理に合致する命令は 1 つである。このようにして、個数算出処理（S 252）では 6 通りの命令並びの各々についてデフォルト論理に合致する命令の個数が求められ、それぞれ 3、1、1、0、0 および 1 となる。命令並び選択処理（S 254）では、6 通りの命令並びのうち、デフォルト論理に合致する命令数の最大値をとる図 47（a）に示す命令並びが選択される。

【0180】

以上のように、本実施の形態におけるコンパイラ 200 によれば、同一スロットのサイクル間で命令、オペコードおよびレジスタフィールドのハミング距離が小さくなるように命令の配置の最適化が行なわれる。このため、プロセッサの命令レジスタに保持される値のビット変化が小さく、プロセッサを低消費電力で動作させることができる機械語プログラムが生成される。

【0181】

また、同一スロットの同一レジスタフィールドにおいて、同一のレジスタのアクセスが連続するような命令の配置の最適化が行なわれる。このため、同一のレジスタへのアクセスが連続し、レジスタを選択するための制御信号の変化が少な

くなり、プロセッサを低消費電力で動作させることができる機械語プログラムが生成される。

【0182】

さらに、各スロットにはデフォルト論理に合致するように命令が割り当てられる。このため、同一のスロットでは、プロセッサの同じ構成要素を利用する命令が連続して実行されることになる。このため、プロセッサを低消費電力で動作させることができる機械語プログラムが生成される。

【0183】

さらにまた、1スロットまたは2スロットしか使用しない命令の命令サイクルが連続した場合には、その間、空きスロットへの電力の供給を停止させることができる。このため、プロセッサを低消費電力で動作させることができる機械語プログラムが生成される。

【0184】

さらにまた、プラグマまたはコンパイル時のオプションでプログラム実行時に使用するスロット数を指定することができる。このため、空きスロットを発生させることができ、空きスロットへの電力の供給を停止させることができる。このため、プロセッサを低消費電力で動作させることができる機械語プログラムが生成される。

【0185】

以上、本発明に係るコンパイラについて、実施の形態に基づいて説明したが、本発明は、この実施の形態に限定されるものではない。

例えば、図28および図29を参照して説明した命令再スケジューリング部232の最適命令取出し処理(S122)では、同一のレジスタ番号を有するフィールドの個数(S152)、直前に実行される命令とのハミング距離(S156)、スロットのデフォルト論理(S160)の順に優先して最適命令を特定したが、この優先順位はこれに限られるものではなく、その他の優先順位で最適命令を特定してもよい。

【0186】

また、最適命令を特定する際に考慮するハミング距離、スロットのデフォルト

論理等の各種条件も、この実施の形態に限定されるものではない。要するに、本発明に係るコンパイラによりプロセッサが動作した際に、総消費電力量が小さくなるような、条件の組み合わせや、優先順位であればよいのである。なお、命令スケジューリング部 232 以外の、命令スケジューリング部 232、レジスタ割付部 234 およびサイクル内配置調整処理部 237 などの処理においても同様であるのは言うまでもない。

【0187】

さらに、これら条件の組み合わせや、優先順位をパラメータ化し、ソースプログラム 202 のヘッダファイルとしてコンパイル時に組み込む構成としてもよいし、これらのパラメータをコンパイラのオプションとして指定可能としてもよい。

【0188】

さらにまた、本実施の形態の最適化部 230 での処理は、基本ブロックごとにいくつかのスケジューリング方法の中から最適なものを選択するようにしてもよい。例えば、基本ブロックごとに、あらかじめ用意された複数のスケジューリング方法のすべてについてスケジューリング結果を求め、最も消費電力が小さくなると予測されるスケジューリング方法を選択するようにしてもよい。

【0189】

また、バックトラックなどの手法を用いて最適なスケジューリング方法を選択するようにしてもよい。例えば、命令スケジューリング部 232 において最も消費電力が小さくなると予測されるスケジューリング方法を選択した後であっても、レジスタ割付部 234 でレジスタ割付を行なったところ、予測消費電力が予定していた値よりも大きくなった場合には、命令スケジューリング部 232 において 2 番目に消費電力が小さくなると予測されるスケジューリング方法を選択して、レジスタ割付を行なってみる。その結果、予測消費電力が予定していた値よりも小さくなれば、命令再スケジューリング部 236 による命令再スケジューリング処理を実行するようにしてもよい。さらにまた、本実施の形態では、C 言語で記述されたソースプログラムを機械語プログラムに変換する例について説明したが、ソースプログラムは C 言語以外的高级言語であってもよいし、他のコンパイラ

ですすでにコンパイルされた機械語プログラムであってもよい。ソースプログラムが機械語プログラムの場合には、その機械語プログラムを最適化した機械語プログラムが出力される構成となる。

【0190】

【発明の効果】

以上の説明から明らかなように、本発明に係るコンパイラ装置によると、プロセッサの命令レジスタに保持される値のビット変化が小さく、プロセッサを低消費電力で動作させることができる命令列が生成される。

【0191】

また、同一のレジスタへのアクセスが連続し、レジスタを選択するための制御信号の変化が少なくなり、プロセッサを低消費電力で動作させることができる命令列が生成される。

【0192】

さらに、同一スロットで同じ構成要素を利用する命を連続して実行させることができるため、プロセッサを低消費電力で動作させることができる命令列が生成される。

【0193】

さらにまた、空きスロットへの電力の供給を停止させることができるため、プロセッサを低消費電力で動作させることができる命令列が生成される。

以上のように、本発明に係るコンパイラにより、並列処理可能なプロセッサを低消費電力で動作させることが可能となる。特に、携帯電話、情報携帯端末等の携帯型の情報処理装置のように、低消費電力での動作が求められる装置で用いられるプロセッサ向きの命令列（機械語プログラム）を生成することができ、その実用的価値は極めて高い。

【図面の簡単な説明】

【図1】 本実施の形態に係るプロセッサが解読実行する命令の構造を示す図である。

【図2】 本実施の形態に係るプロセッサの概略構成を示すブロック図である。

【図 3】 パケットの一例を示す図である。

【図 4】 パケットに含まれる並列実行境界情報について説明するための図である。

【図 5】 パケットの並列実行境界情報に基づいて作成された並列実行される命令の実行単位の一例を示す図である。

【図 6】 算術論理・比較演算部の概略構成を示すブロック図である。

【図 7】 バレルシフタの概略構成を示すブロック図である。

【図 8】 除算器の概略構成を示すブロック図である。

【図 9】 乗算・積和演算部の概略構成を示すブロック図である。

【図 10】 プロセッサによる命令実行時の各パイプライン動作を示すタイミング図である。

【図 11】 プロセッサで実行される命令、処理の内容およびそのビットパターンを示す図である。

【図 12】 本実施の形態に係るコンパイラの構成を示す機能ブロック図である。

【図 13】 命令スケジューリング部の動作を示すフローチャートである。

【図 14】 依存グラフの一例を示す図である。

【図 15】 命令のスケジュール結果の一例を示す図である。

【図 16】 図 13 の最適命令取出し処理の動作を示すフローチャートである。

【図 17】 オペコードのビットパターン間のハミング距離の算出方法を説明するための図である。

【図 18】 ビット長が異なるオペコード間でのハミング距離の算出方法を説明するための図である。

【図 19】 サイクル内配置調整処理部の動作を示すフローチャートである。

【図 20】 6 通りの命令並びの一例を示す図である。

【図 21】 配置された命令の一例を示す図である。

【図 22】 命令並び作成処理（図 19 の S 6 1）を説明するための図であ

る。

【図 2 3】 オペコードのハミング距離算出処理（図 1 9 の S 6 4）を説明するための図である。

【図 2 4】 レジスタ割付部の動作を示すフローチャートである。

【図 2 5】 割付対象となる変数の生存区間を示す図である。

【図 2 6】 図 2 5 の例に基づいて作成された変数の干渉グラフを示す図である。

【図 2 7】 命令スケジューリングの途中結果を示す図である。

【図 2 8】 命令再スケジューリング部の動作を示すフローチャートである。

【図 2 9】 図 2 8 の最適命令取出し処理の動作を示すフローチャートである。

【図 3 0】 配置候補命令特定処理（図 2 9 の S 1 5 2）を説明するための図である。

【図 3 1】 配置候補命令特定処理（図 2 9 の S 1 5 6）を説明するための図である。

【図 3 2】 配置候補命令特定処理（図 2 9 の S 1 6 0）を説明するための図である。

【図 3 3】 スロット停止・復帰命令生成部の動作を示すフローチャートである。

【図 3 4】 命令が配置されたスケジュール結果の一例を示す図である。

【図 3 5】 図 3 3 の特定の 1 スロットのみが連続使用されている場合の処理で命令が書き込まれたスケジュール結果の一例を示す図である。

【図 3 6】 図 3 3 の特定の 2 スロットのみが連続使用されている場合の処理で命令が書き込まれたスケジュール結果の一例を示す図である。

【図 3 7】 プログラム状態レジスタの一例を示す図である。

【図 3 8】 プログラム状態レジスタの他の一例を示す図である。

【図 3 9】 図 2 8 の最適命令取出し処理の他の動作を示すフローチャートである。

【図 4 0】 配置候補命令特定処理（図 3 9 の S 2 1 2）を説明するための図である。

【図 4 1】 サイクル内配置調整処理部 2 3 7 の動作の第 1 変形例を示すフローチャートである。

【図 4 2】 命令のハミング距離算出処理（図 4 1 の S 2 2 2）を説明するための図である。

【図 4 3】 サイクル内配置調整処理部 2 3 7 の動作の第 2 変形例を示すフローチャートである。

【図 4 4】 レジスタフィールドのハミング距離算出処理（図 4 3 の S 2 3 2）を説明するための図である。

【図 4 5】 サイクル内配置調整処理部 2 3 7 の動作の第 3 変形例を示すフローチャートである。

【図 4 6】 配置された命令の一例を示す図である。

【図 4 7】 命令並び作成処理（図 4 5 の S 6 1）を説明するための図である。

【図 4 8】 レジスタフィールド個数算出処理（図 4 5 の S 2 4 2）を説明するための図である。

【図 4 9】 サイクル内配置調整処理部 2 3 7 の動作の第 4 変形例を示すフローチャートである。

【符号の説明】

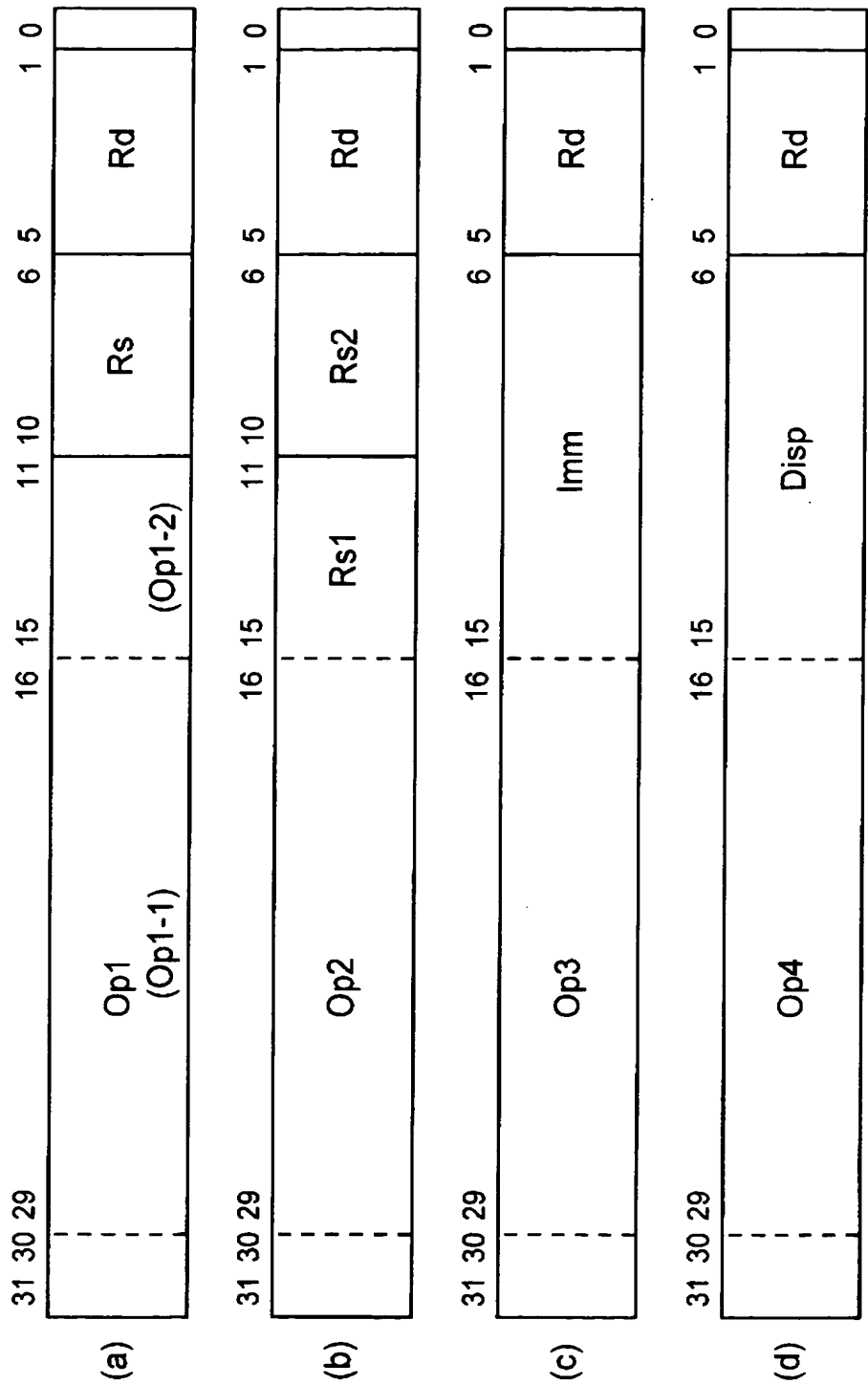
- 3 0 プロセッサ
- 4 0 命令メモリ
- 5 0 命令供給発行部
- 5 2 命令フェッチ部
- 5 4 命令バッファ
- 5 6 命令レジスタ部
- 5 6 a ～ 5 6 c 命令レジスタ
- 6 0 解読部
- 6 2 命令発行制御部

64 デコード部
64a～64c 命令デコーダ
70 実行部
72 実行制御部
74 PC部
76 レジスタファイル
78a～78c 算術論理・比較演算部
80a, 80b 乗算・積和演算部
82a～82c バレルシフタ
84 除算器
88 オペランドアクセス部
90, 92 データバス
94 ODバス
96 OAバス
100 データメモリ
112, 114 パケット
122, 124, 126 実行単位
132 ALU部
134, 154, 180 飽和处理部
136 フラグ部
142, 162, 172 アキュムレータ部
146, 148, 178 セレクタ
150 上位バレルシフタ
152 下位バレルシフタ
164 除算部
174a, 174b 32ビット乗算器 (MUL)
176a～176c 64ビット加算器 (Adder)
200 コンパイラ
202 ソースプログラム

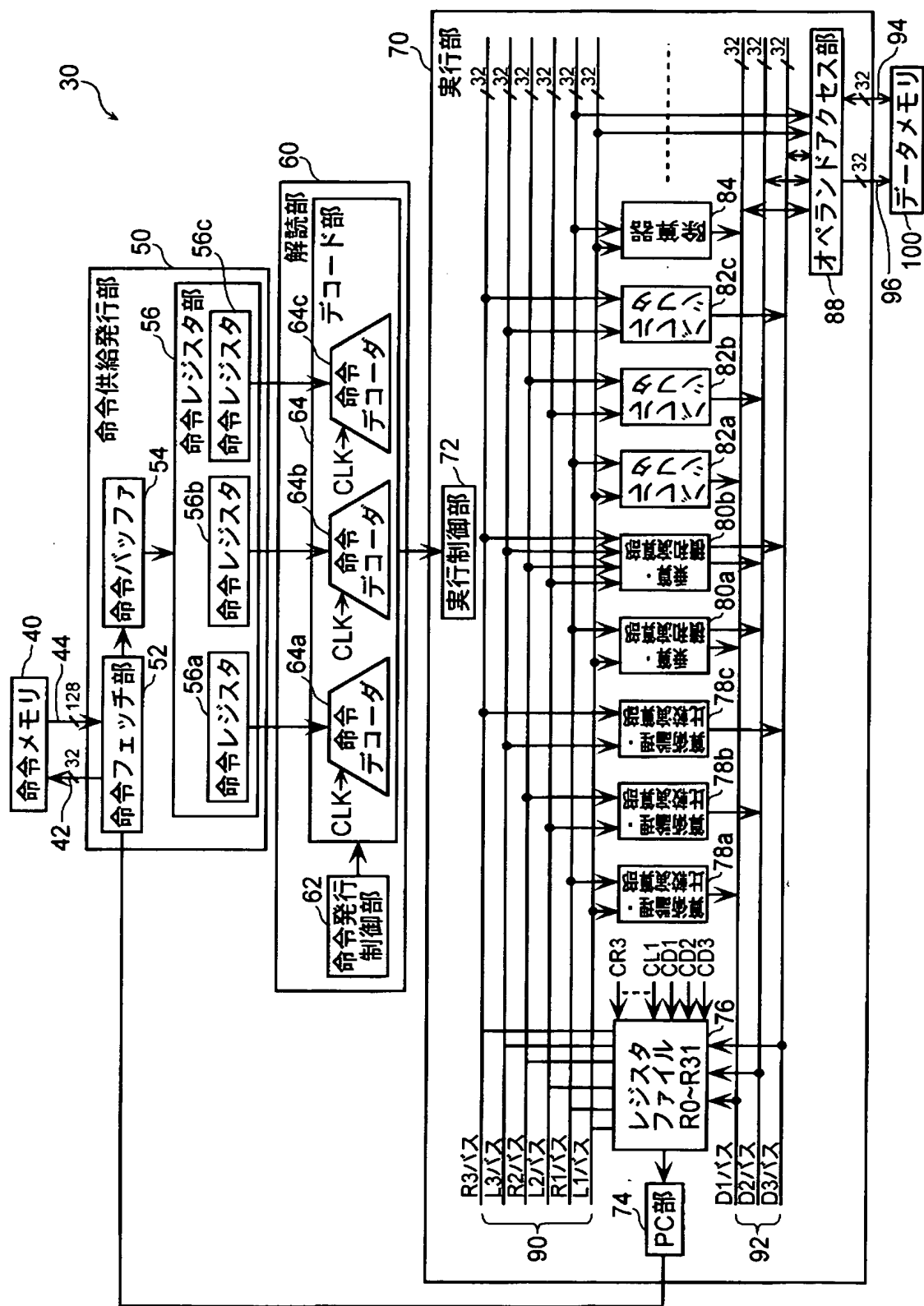
2 0 4	機械語プログラム
2 1 0	パーサー部
2 2 0	中間コード変換部
2 3 0	最適化部
2 3 2	命令スケジューリング部
2 3 4	レジスタ割付部
2 3 6	命令再スケジューリング部
2 3 7	サイクル内配置調整処理部
2 3 8	スロット停止・復帰命令生成部
2 3 9	並列実行境界情報設定部
2 4 0	コード生成部

【書類名】 図面

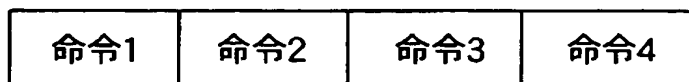
【図 1】



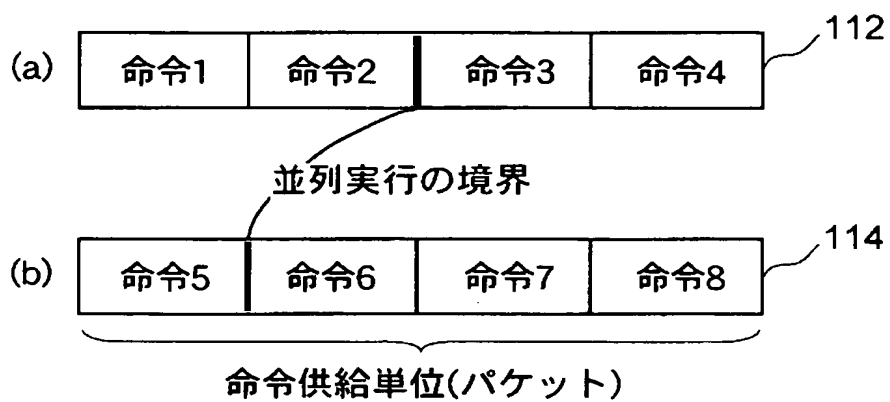
【図 2】



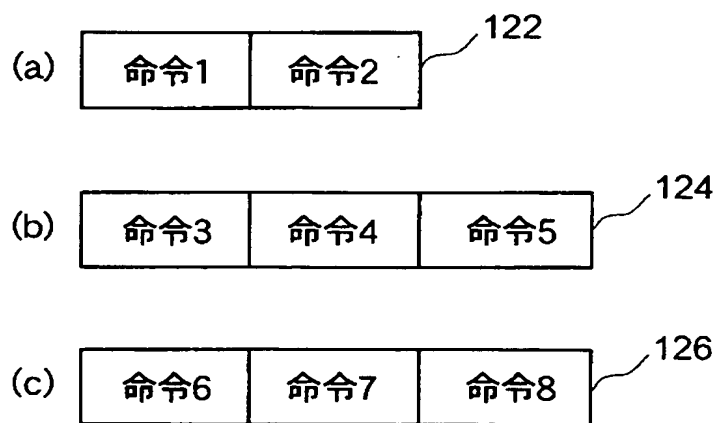
【図 3】



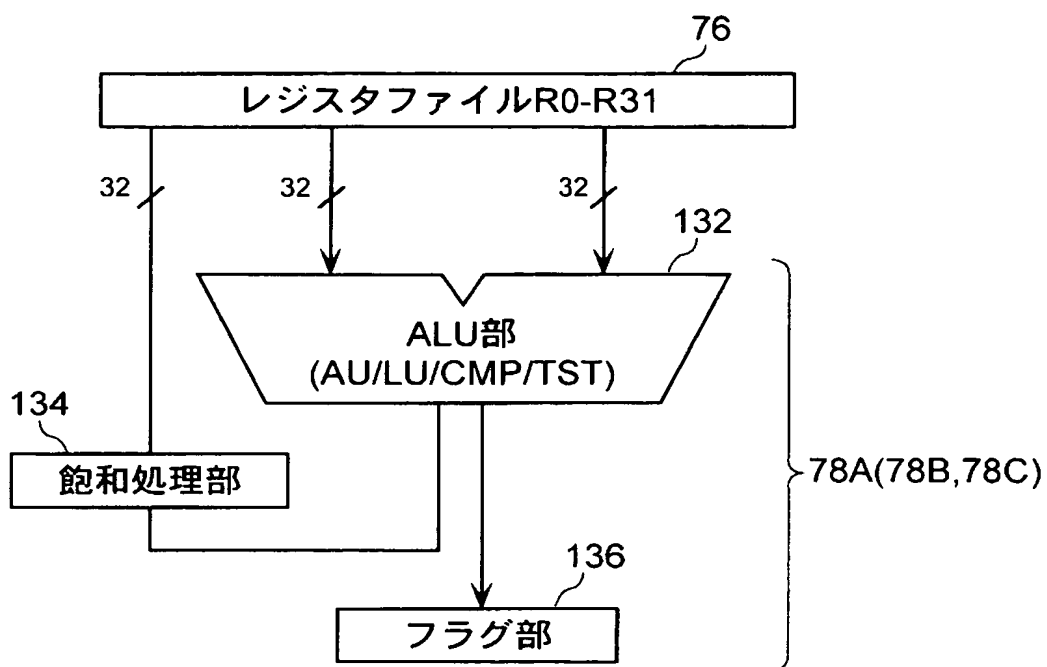
【図 4】



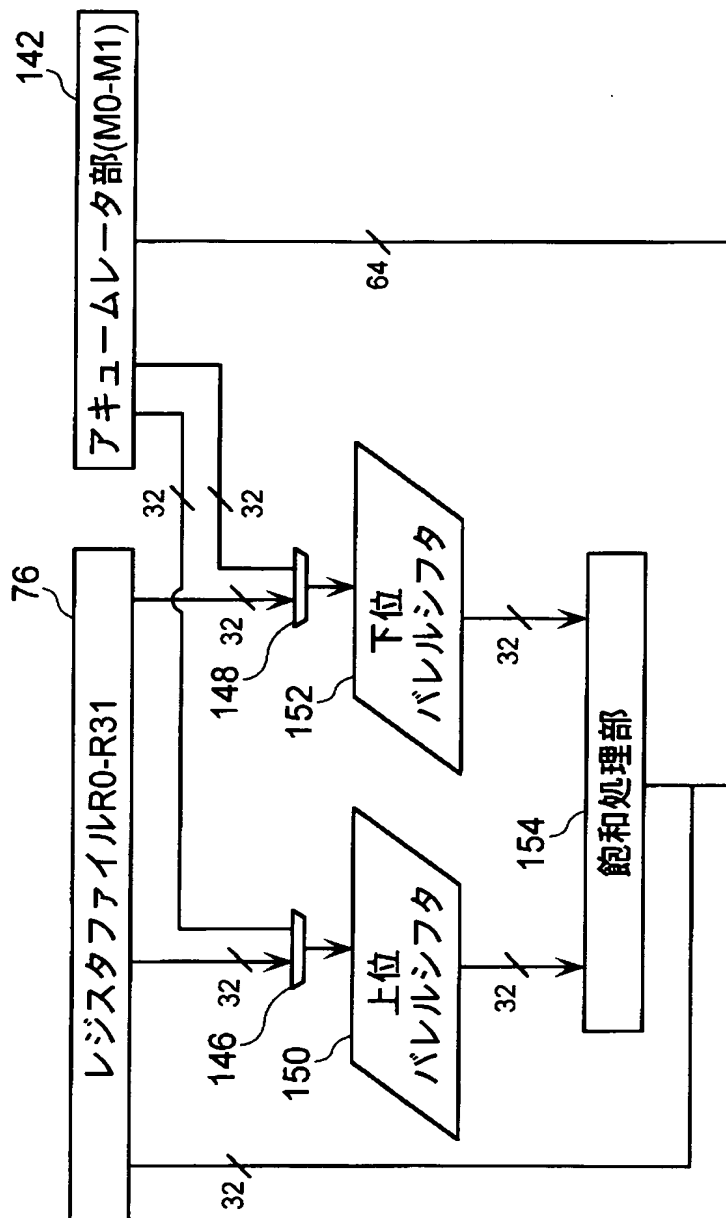
【図 5】



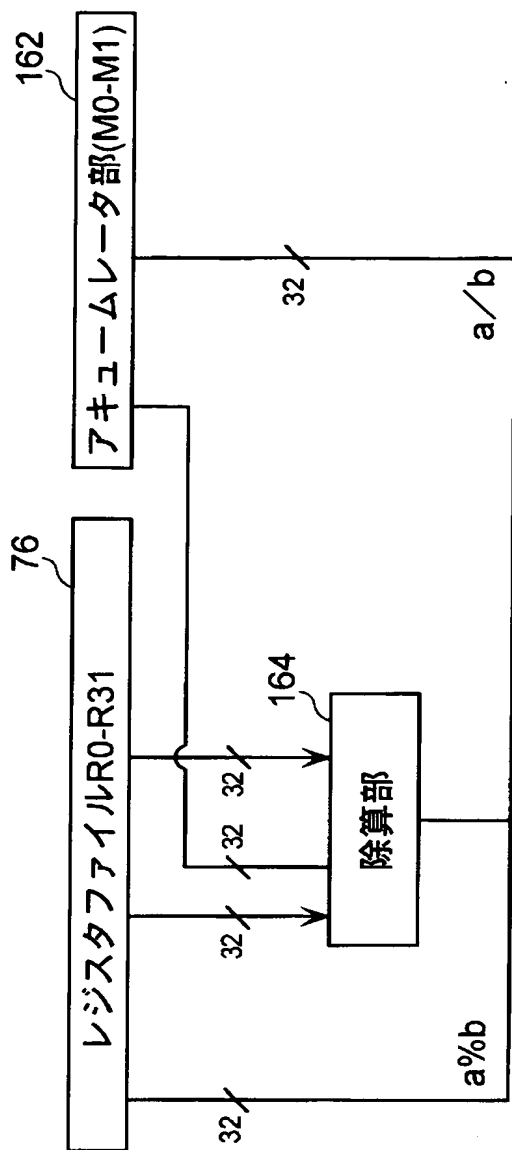
【図 6】



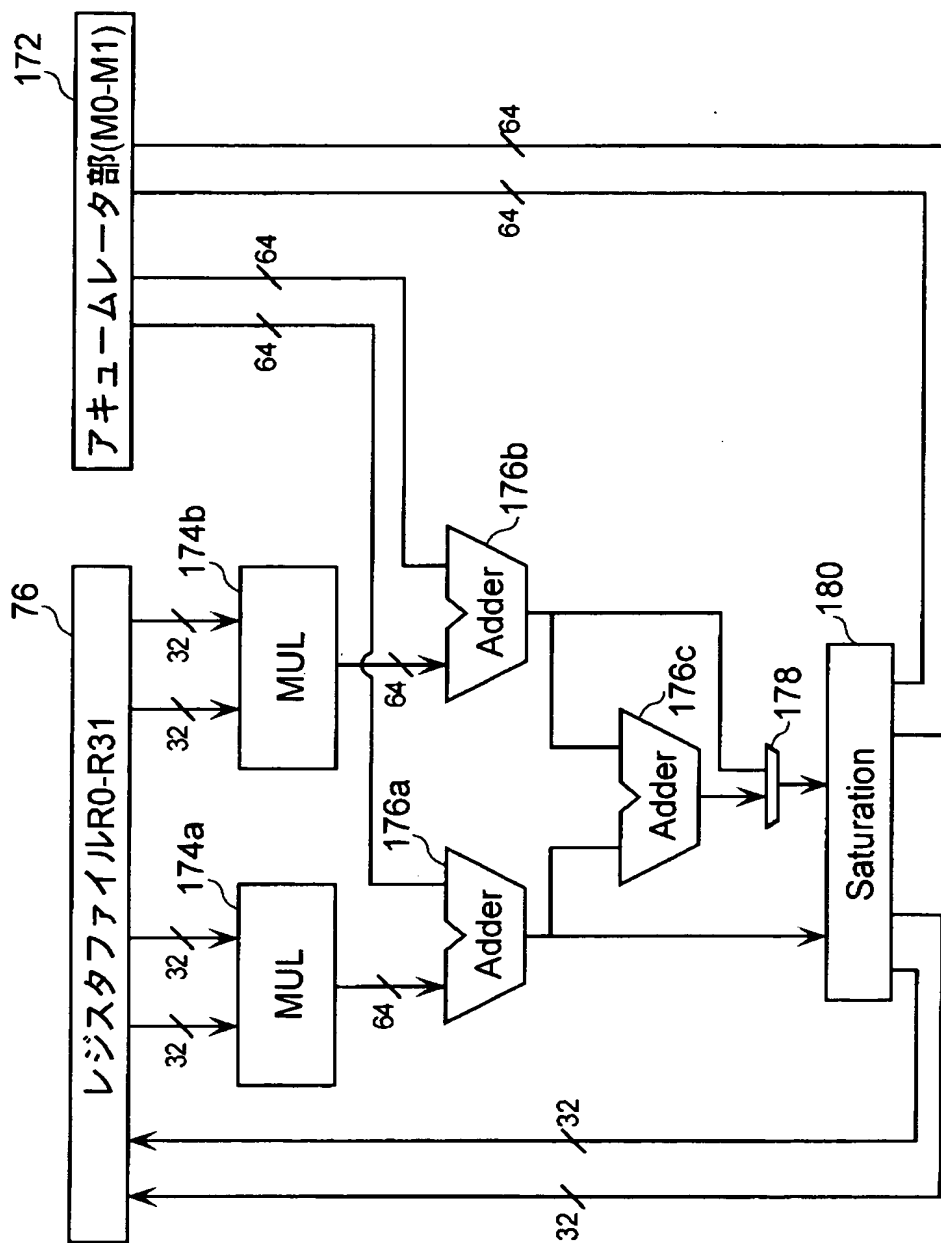
【図 7】



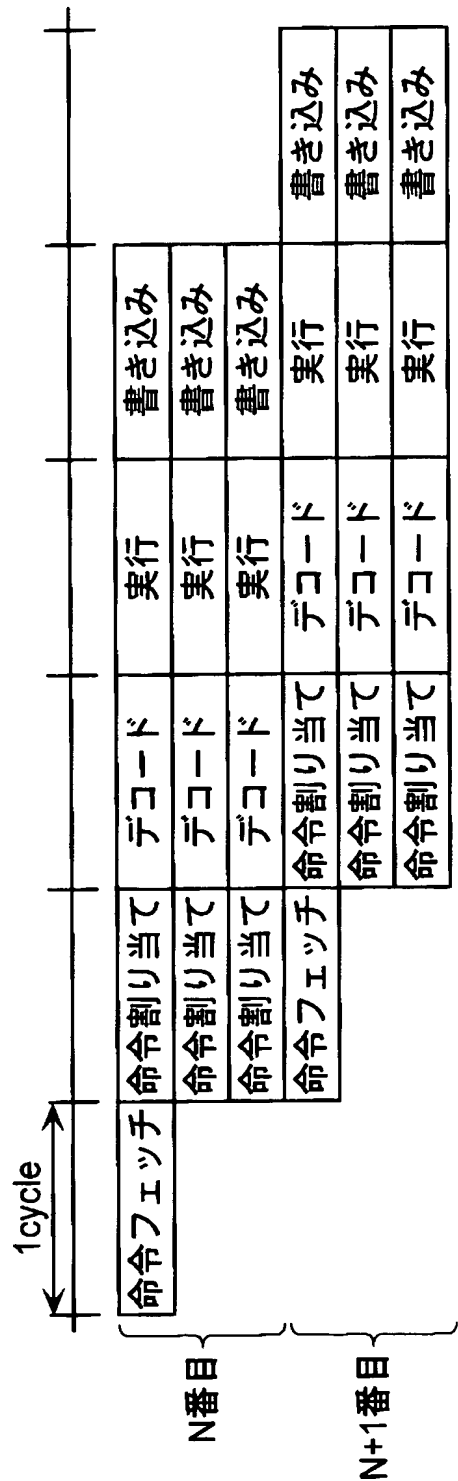
【図 8】



【図 9】



【図 10】



【図 11】

命令	処理の内容	31	30	29	16	15	11	10	6	5	1	0
ld Rs, Rd	(Rs)→Rd	0	1	0	1	0	1	1	0	0	0	1
st Rs, Rd	(Rs)→(Rd)	0	1	0	1	0	1	0	1	0	0	1
mul1 Rs, Rd	(Rs)×(Rd)→Rd	1	0	0	1	0	0	1	1	0	0	1
mul2 Rs1, Rs2, Rd	(Rs1)×(Rs2)→Rd	1	0	0	0	1	0	0	1	1	0	0
add1 Rs, Rd	(Rs)+(Rd)→Rd	1	1	0	0	0	0	0	1	0	0	1
add2 Rs1, Rs2, Rd	(Rs1)+(Rs2)→Rd	1	1	0	0	0	0	0	1	1	0	1
sub1 Rs, Rd	(Rs)-(Rd)→Rd	1	1	0	0	0	0	0	1	0	0	1
sub2 Rs1, Rs2, Rd	(Rs1)-(Rs2)→Rd	1	1	0	0	0	0	0	1	1	0	1
mov1 Rs, Rd	(Rs)→Rd	1	1	0	0	0	0	1	1	0	0	1
mov2 Imm, Rd	Imm→Rd	1	1	0	0	0	1	1	0	0	1	0
div Rs, Rd	(Rs)÷(Rd)→Rd	1	1	0	0	1	0	0	1	0	0	1
mod Rs, Rd	(Rs)%(Rd)→Rd	1	1	0	0	1	0	0	1	1	0	1

オペレーション群

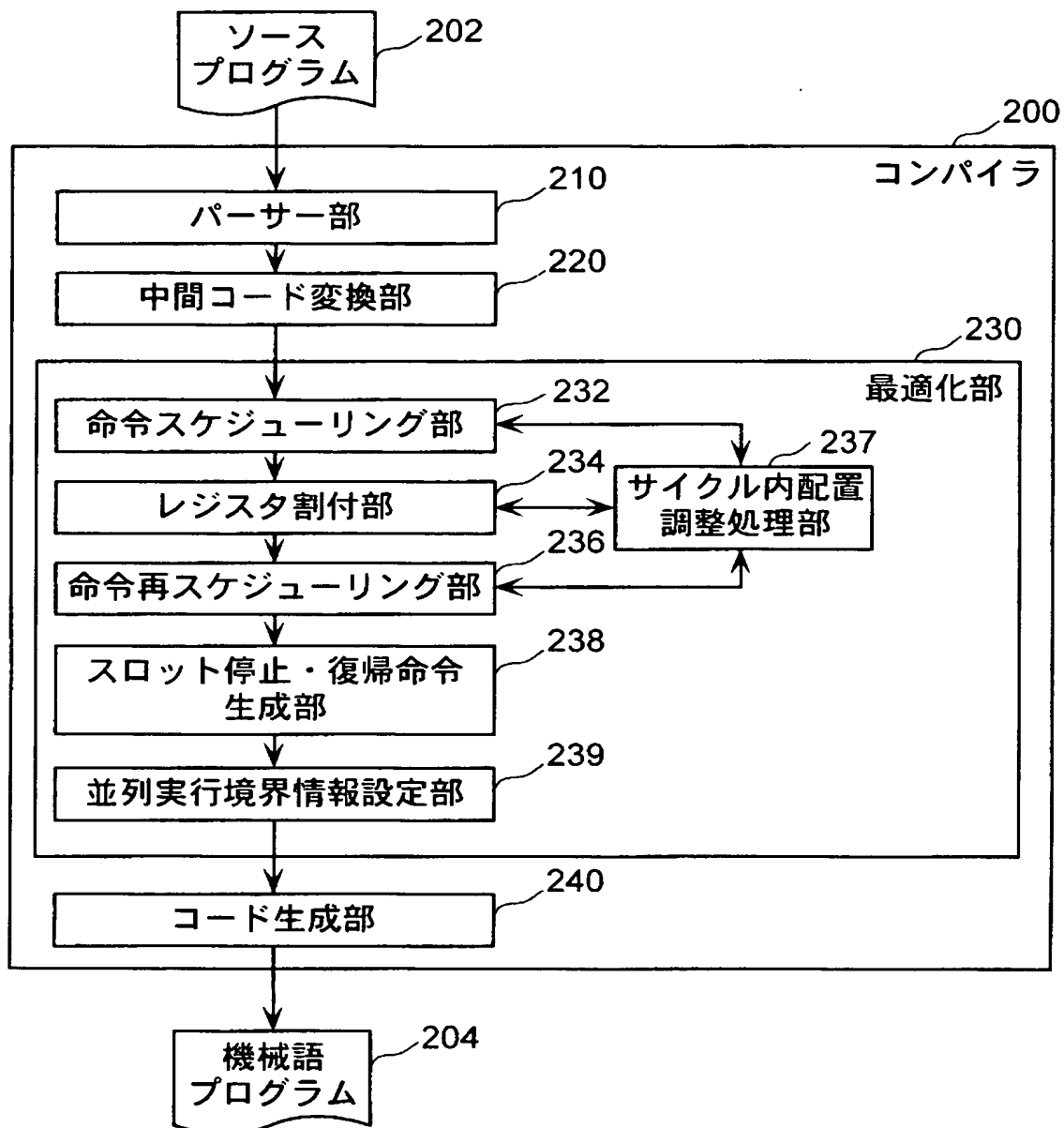
Op1-1, Op2, Op3, Op4

Op1

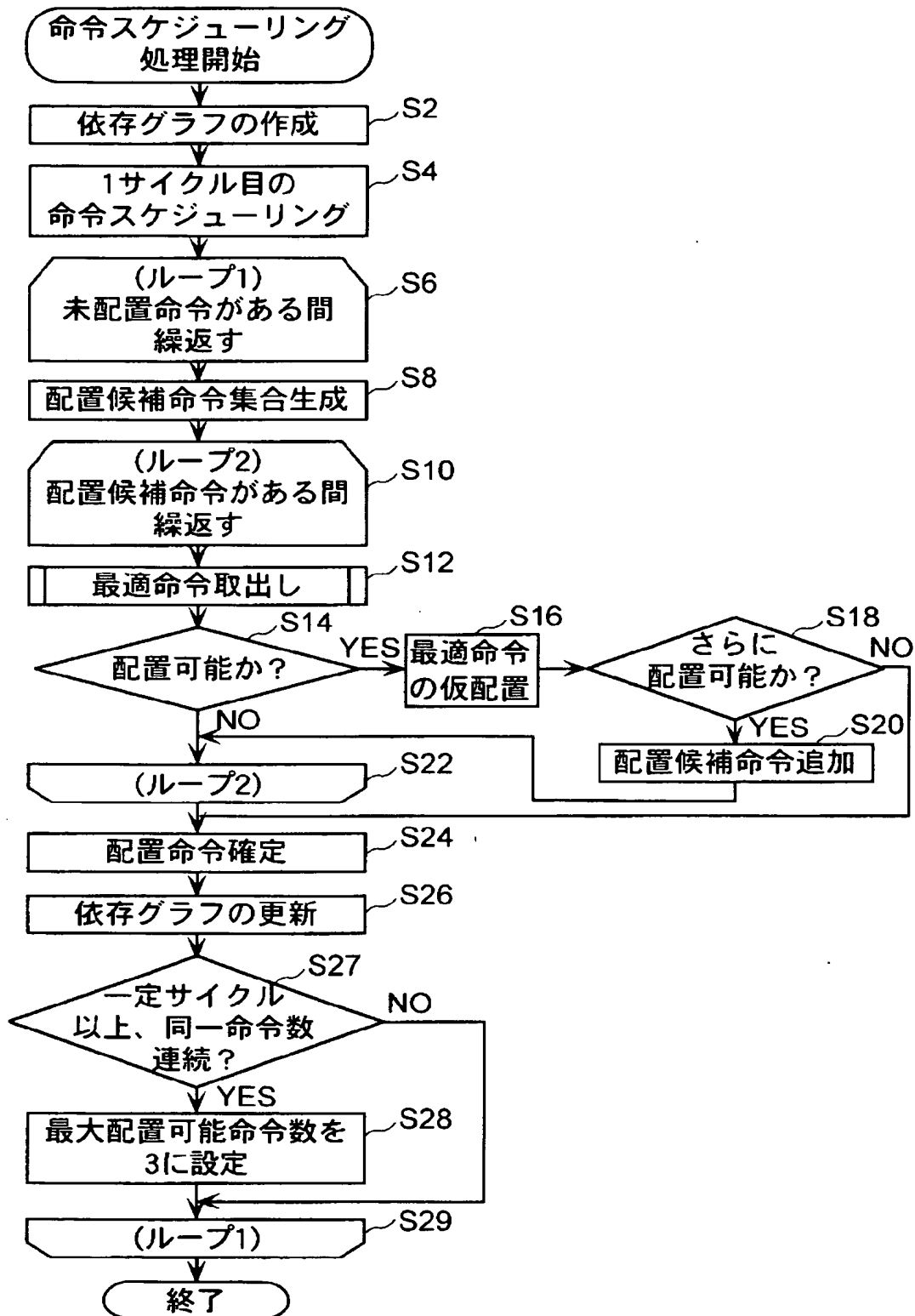
Op1-2

並列実行境界情報

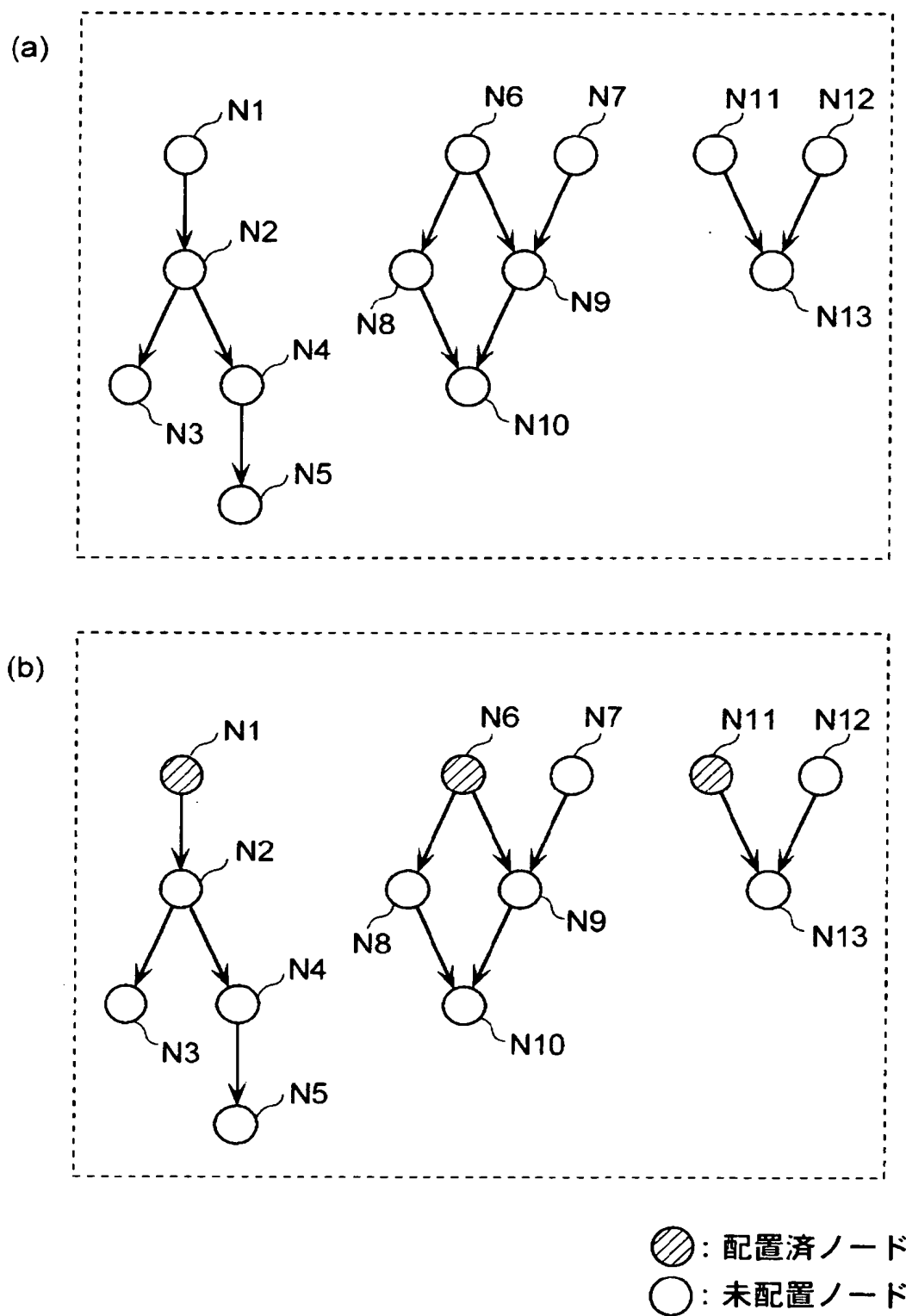
【図 12】



【図13】



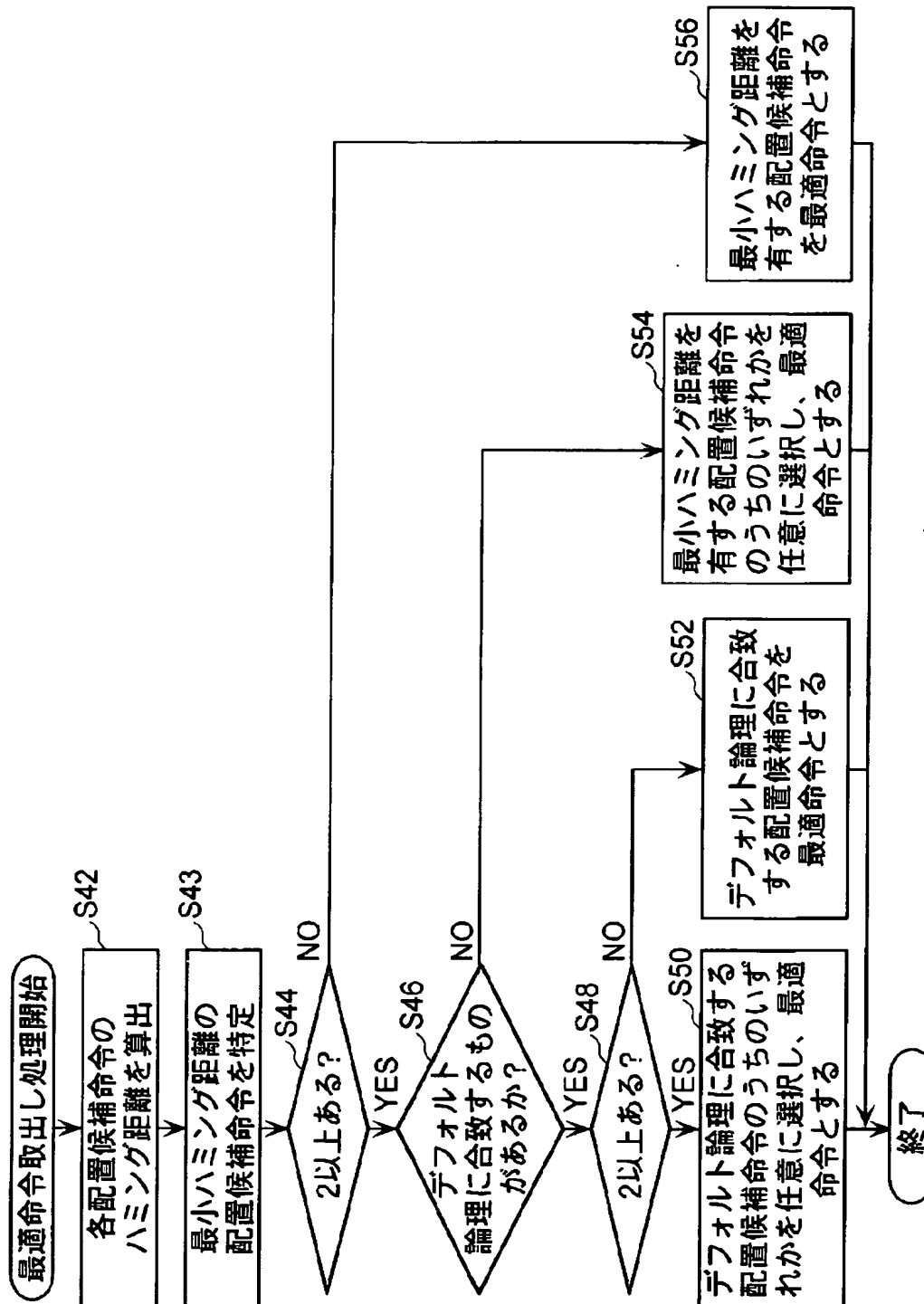
【図 14】



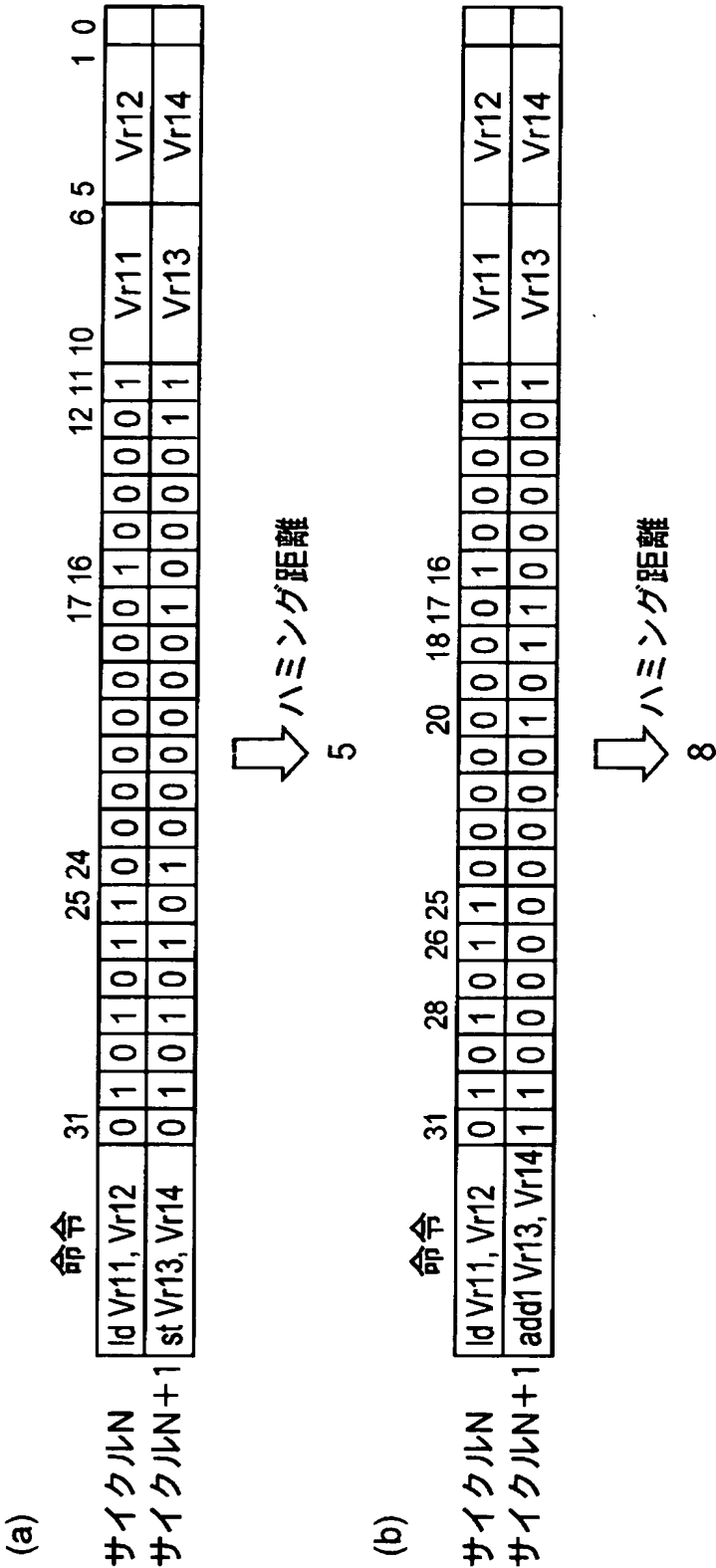
【図 15】

サイクル	第1スロット	第2スロット	第3スロット
1	ld Vr0, Vr1 (ノードN1)	mul2 Vr2, Vr3, Vr4 (ノードN11)	add1 Vr5, Vr6 (ノードN6)

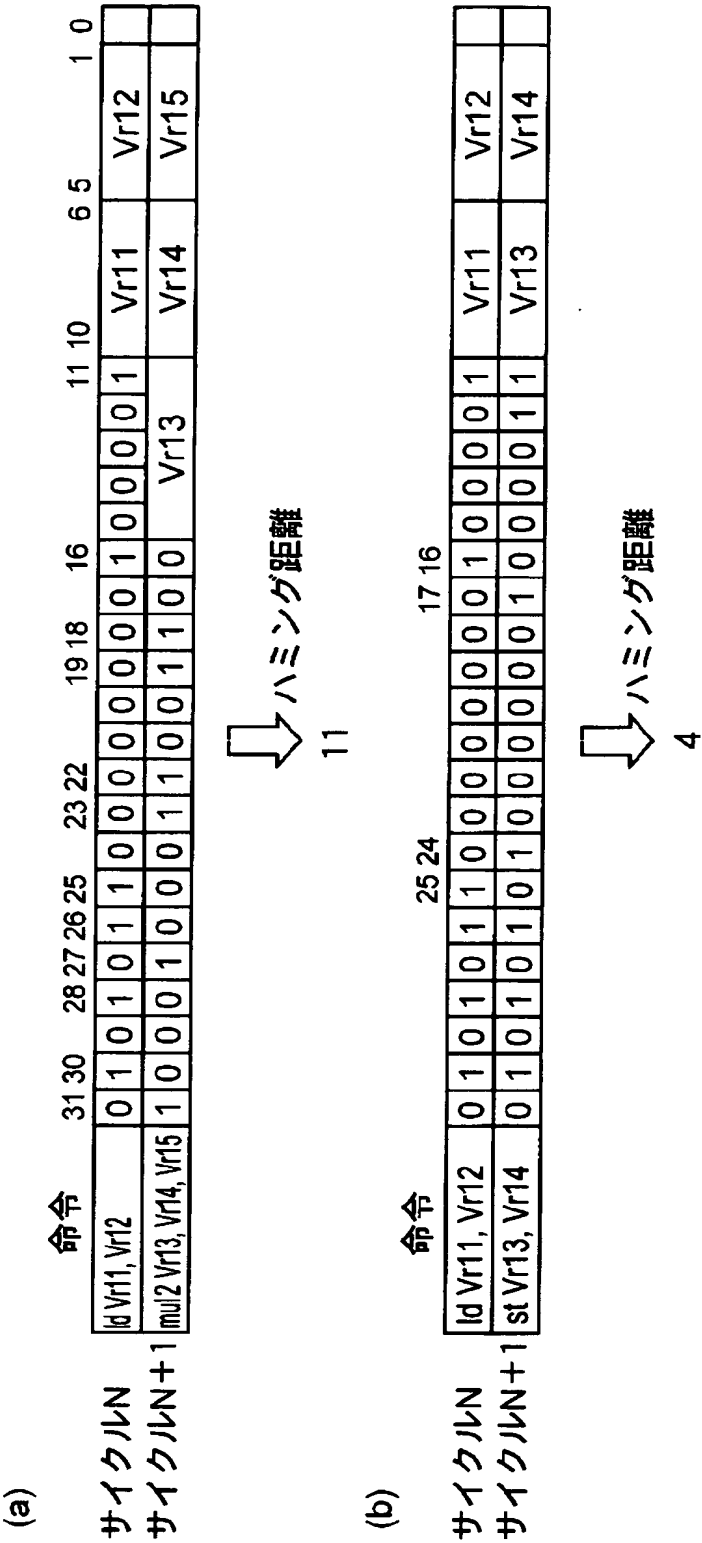
【図 16】



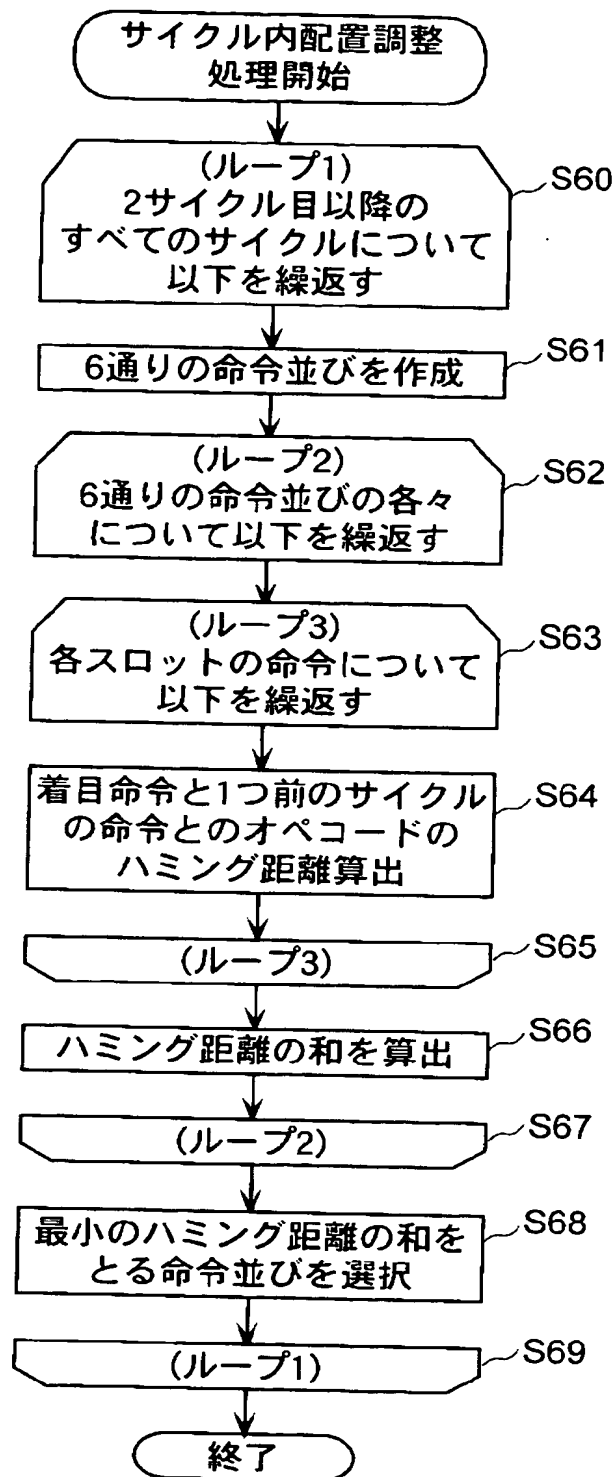
【図 1 7】



【図 18】



【図19】



【図 20】

- (a)

命令1	命令2	命令3
-----	-----	-----
- (b)

命令1	命令3	命令2
-----	-----	-----
- (c)

命令2	命令1	命令3
-----	-----	-----
- (d)

命令2	命令3	命令1
-----	-----	-----
- (e)

命令3	命令1	命令2
-----	-----	-----
- (f)

命令3	命令2	命令1
-----	-----	-----

【図 2 1】

サクル	第1スロット	第2スロット	第3スロット
- - -	- - -	- - -	- - -
N	0101011000000000100001 id :Vr10:Vr11	1100000001000101000001 sub1 :Vr12:Vr13	1100000000010110000001 add1 :Vr14:Vr15
N+1	01010101000000010000011 st :Vr16:Vr17	100010001100010000000001 mul1 :Vr18:Vr19	110010001101000000000001 mod :Vr20:Vr21
- - -	- - -	- - -	- - -

【図22】

(a)	サイクル	第1スロット	第2スロット	第3スロット
N+1	st	Vr16 : Vr17 010101010000001000011	mul1 1000100011000100000001	mod 1100100011010000000001 Vr20 : Vr21
(b)	サイクル	第1スロット	第2スロット	第3スロット
N+1	st	Vr16 : Vr17 010101010000001000011	mod 1100100011010000000001	mul1 1000100011000100000001 Vr18 : Vr19
(c)	サイクル	第1スロット	第2スロット	第3スロット
N+1	mul1	Vr18 : Vr19 1000100011000100000001	st 010101010000001000011	mod 1100100011010000000001 Vr20 : Vr21
(d)	サイクル	第1スロット	第2スロット	第3スロット
N+1	mul1	Vr18 : Vr19 1000100011000100000001	mod 1100100011010000000001	st 010101010000001000011 Vr16 : Vr17
(e)	サイクル	第1スロット	第2スロット	第3スロット
N+1	mod	Vr20 : Vr21 1100100011010000000001	st 010101010000001000011	mul1 1000100011000100000001 Vr18 : Vr19
(f)	サイクル	第1スロット	第2スロット	第3スロット
N+1	mod	Vr20 : Vr21 1100100011010000000001	mul1 1000100011000100000001	st 010101010000001000011 Vr16 : Vr17

【図 23】

サイクル	第1スロット	第2スロット	第3スロット
N	<div>Id</div> <div>010101100000000100001</div> <div>Vr10:Vr11</div>	<div>sub1</div> <div>110000000100010100001</div> <div>Vr12:Vr13</div>	<div>add1</div> <div>110000000001011000001</div> <div>Vr14:Vr15</div>
N+1	<div>mul1</div> <div>1000100011000100000001</div> <div>Vr18:Vr19</div>	<div>st</div> <div>0101010100000001000011</div> <div>Vr16:Vr17</div>	<div>mod</div> <div>1100100011010000000001</div> <div>Vr20:Vr21</div>

オペコードの
ハミング距離

10

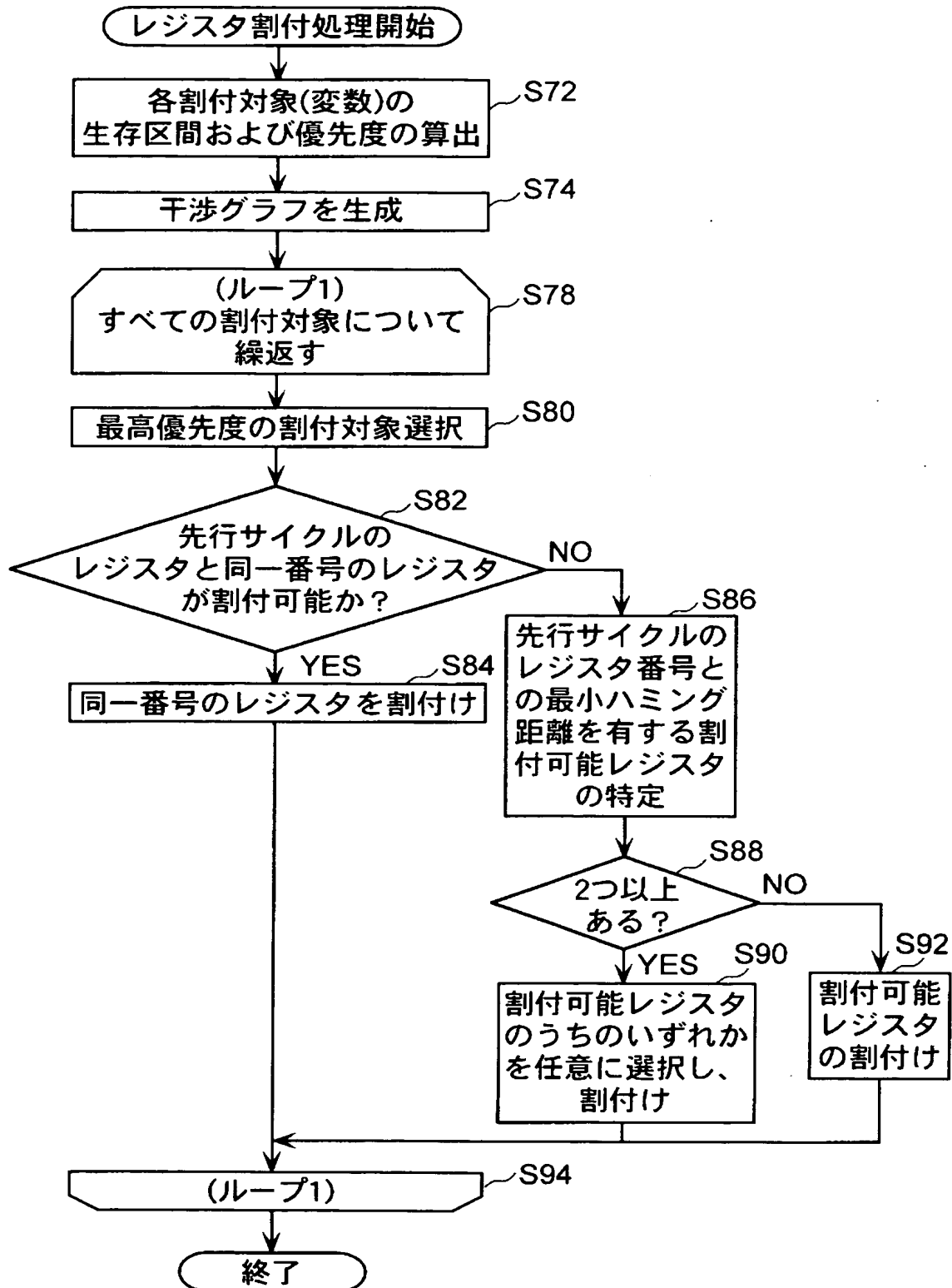
オペコードの
ハミング距離

9

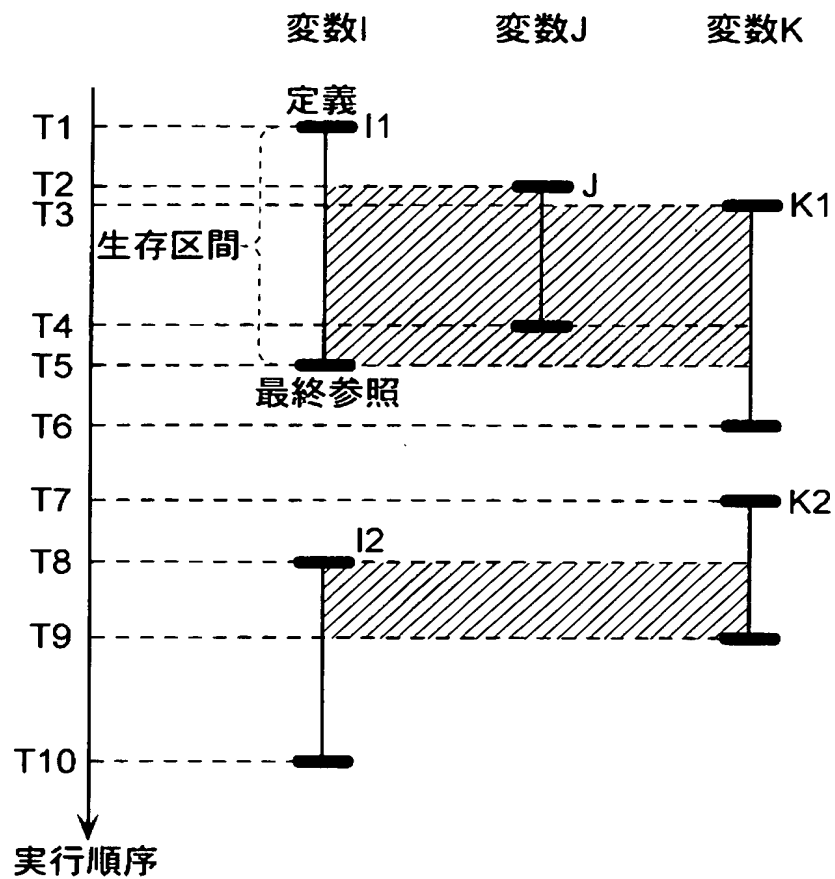
オペコードの
ハミング距離

5

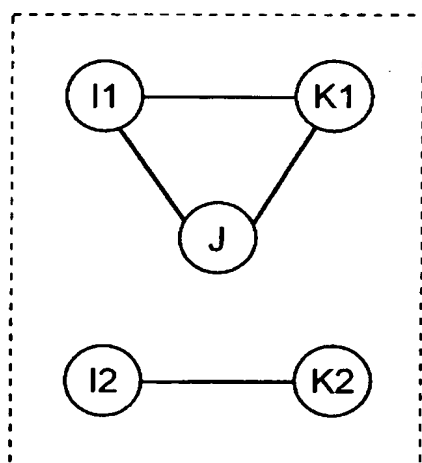
【図 24】



【図 2 5】



【図 2 6】



【図27】

サイクル	第1スロット	第2スロット	第3スロット
⋮	⋮	⋮	⋮
N	add1 R0, R2		
N+1	sub1 Vr5, Vr6		
⋮	⋮		

(a)

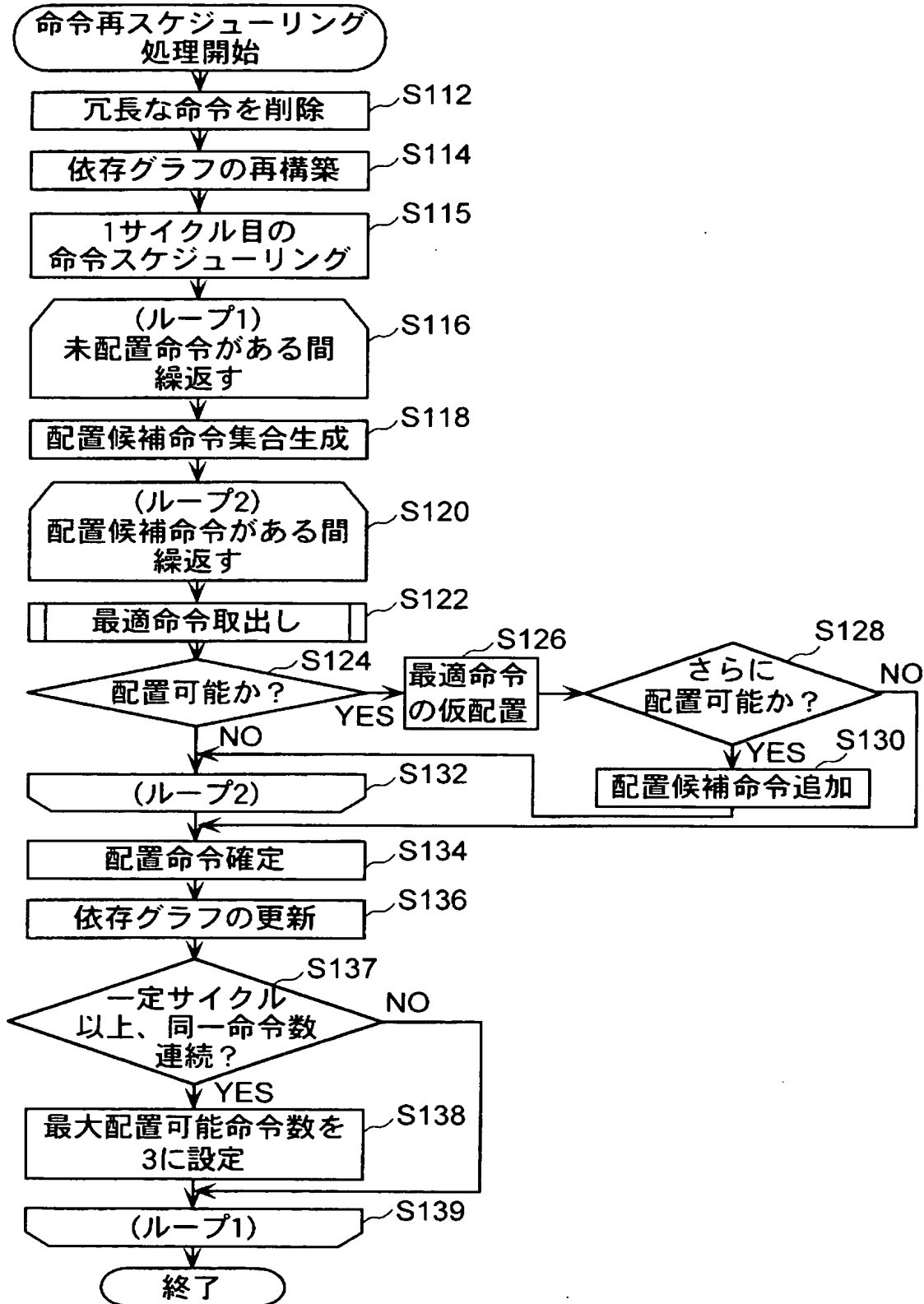
N	add1 11000000001011000001	R0 00000	R2 00010
N+1	sub1 110000000100010100001	R0 00000	Vr6

(b)

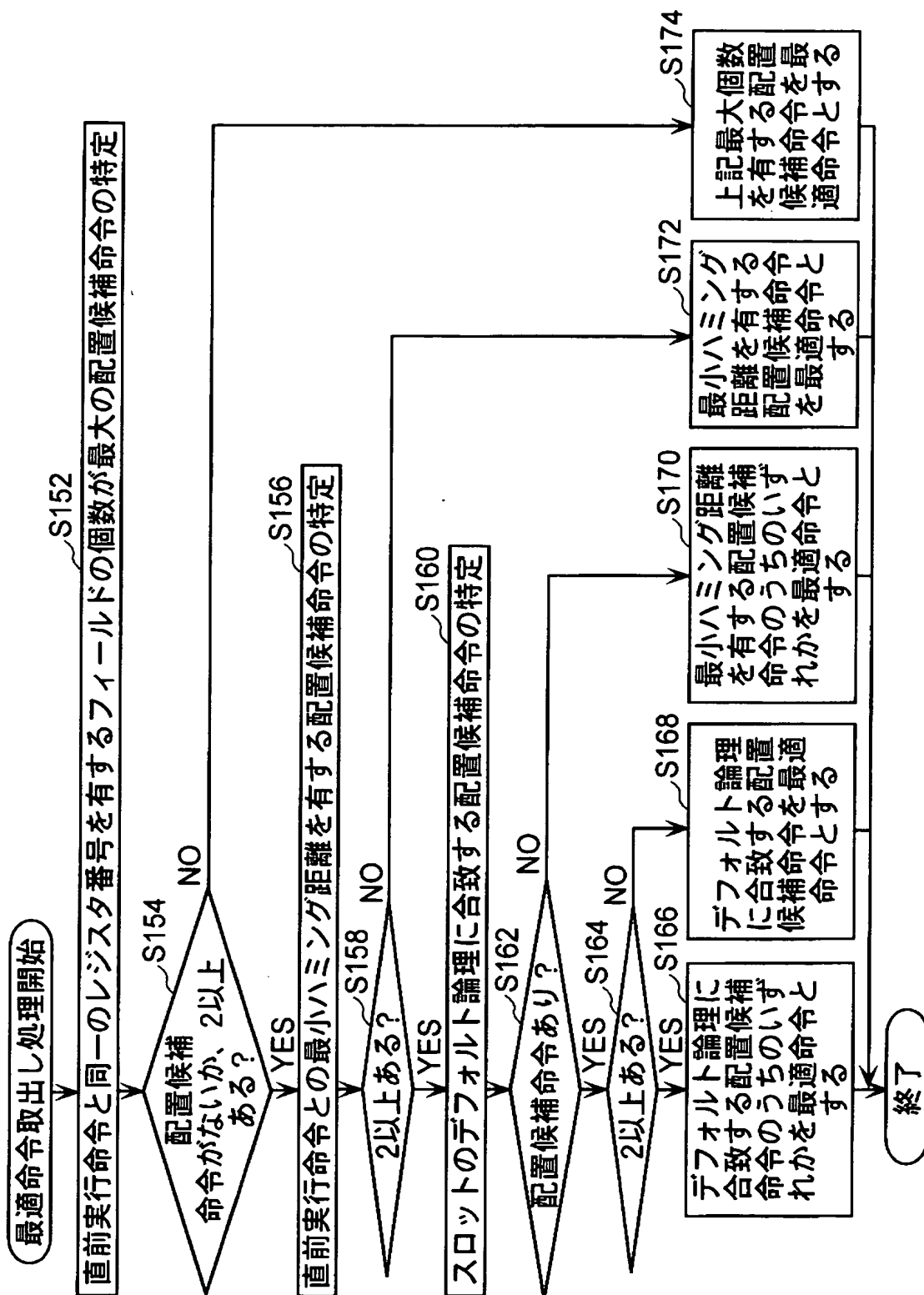
N	add1 110000000001011000001	R0 00000	R2 00010
N+1	sub1 110000000100010100001	R1 00001	Vr6

(c)

【図 28】



【図 29】



【図30】

サイクル	第1スロット		
N	add1 11000000001011000001	R0 00000	R2 00010
N+1	sub1 110000000100010100001	R0 00000	R1 00001

(a)

↓
1
同一レジスタ番号を有する
レジスタフィールドの個数

N	add1 110000000001011000001	R0 00000	R2 00010
N+1	div 1100010010100000000001	R0 00000	R2 00010

(b)

↓
2
同一レジスタ番号を有する
レジスタフィールドの個数

【図 3 1】

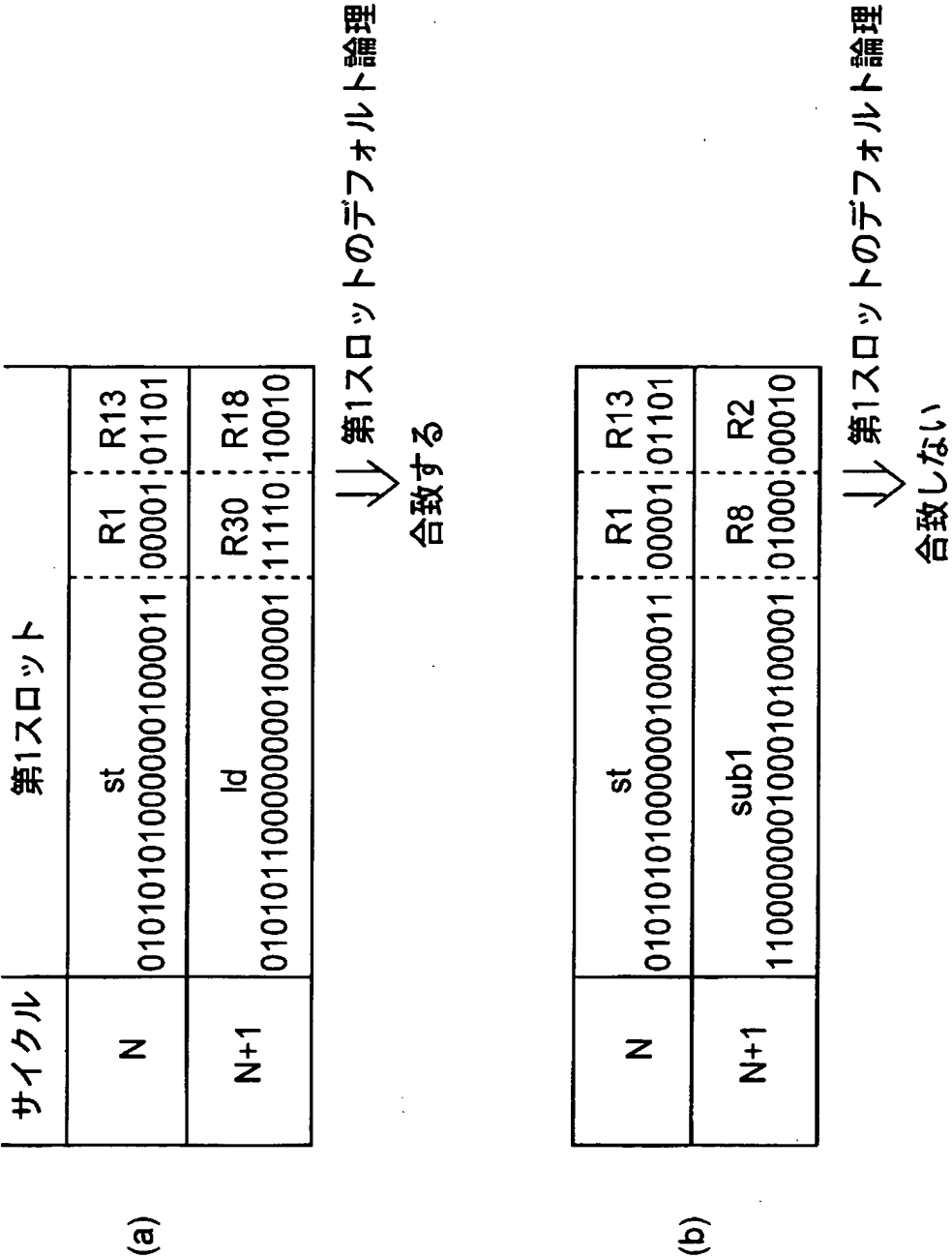
第1スロット			
サイクル			
N	mul1 1000100011000100000001	R3 00011	R10 01010
N+1	add1 11000000001011000001	R2 00010	R4 00100

命令のハミング距離
10

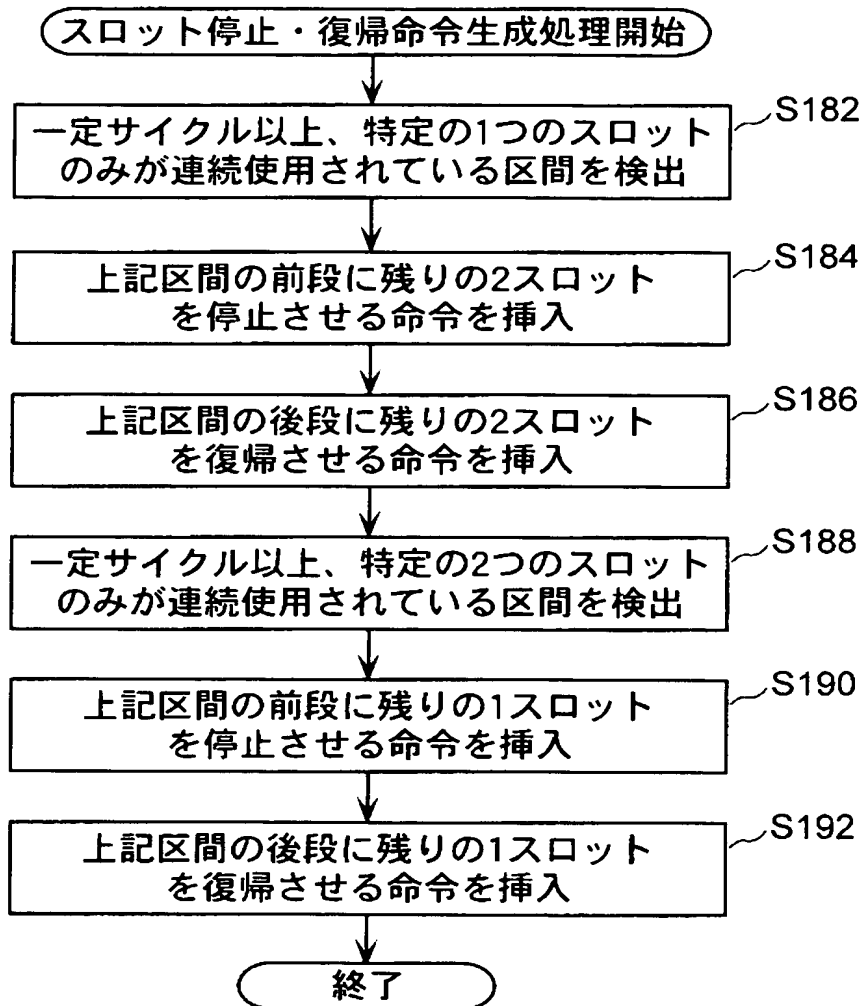
N	mul1 1000100011000100000001	R3 00011	R10 01010
N+1	sub2 1100000011000101010110000000010	R11 01011	R0 00000

命令のハミング距離
8

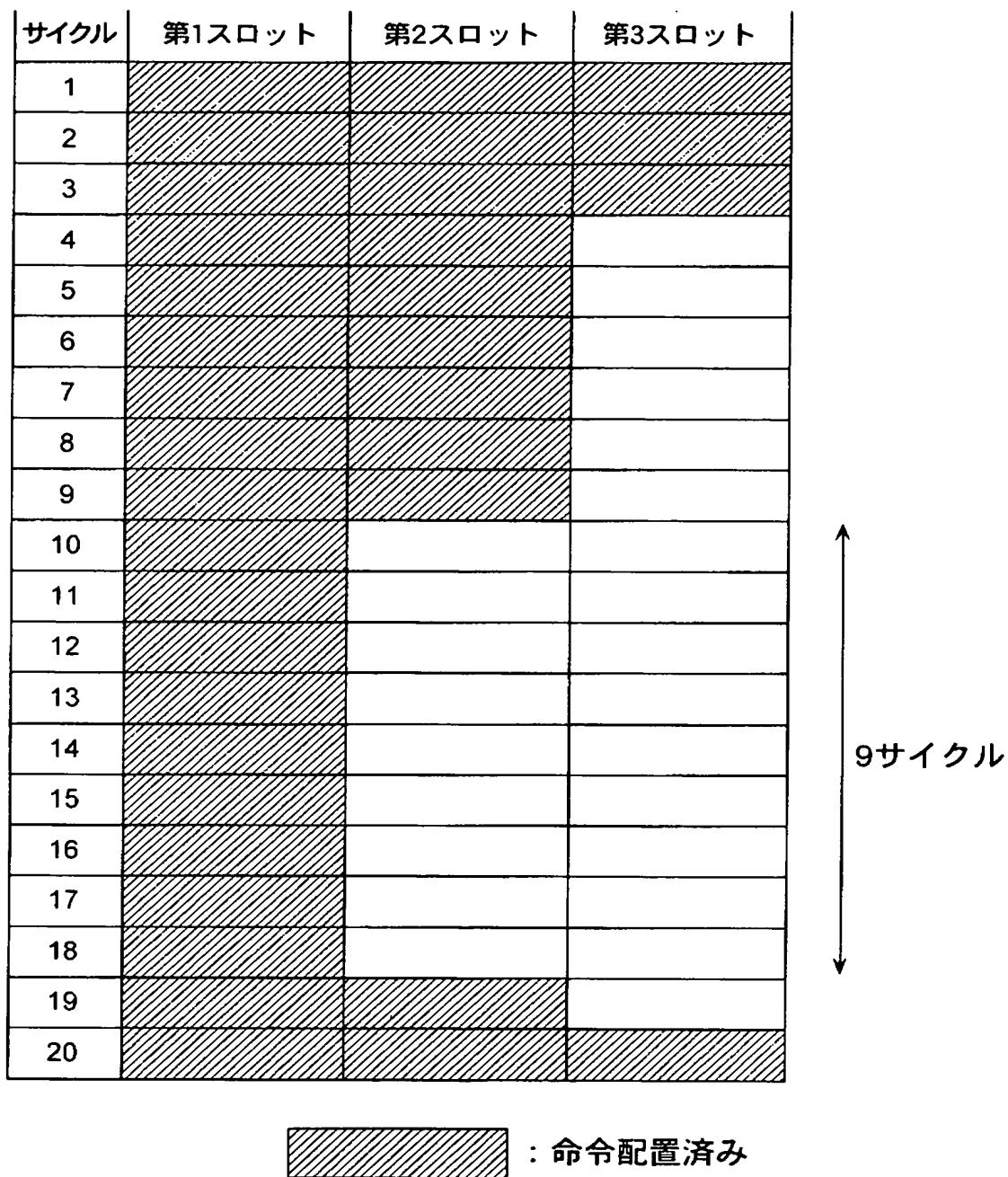
【図 3 2】



【図 33】



【図 3 4】

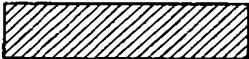


【図35】

サイクル	第1スロット	第2スロット	第3スロット
1			
2			
3			
4			
5			
6			
7			
8			
9			set1 1
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			clear1 1
20			

5サイクル

9サイクル


 : 命令配置済み

【図 36】

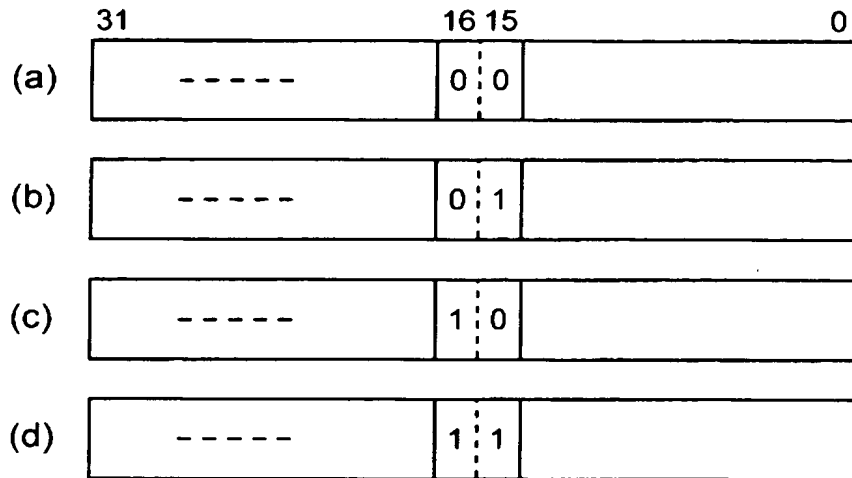
サイクル	第1スロット	第2スロット	第3スロット
1			
2			
3			
4	set2 12		
5			
6			
7			
8			
9			
10	clear2 12		
11			set1 1
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			clear1 1
22			

5サイクル

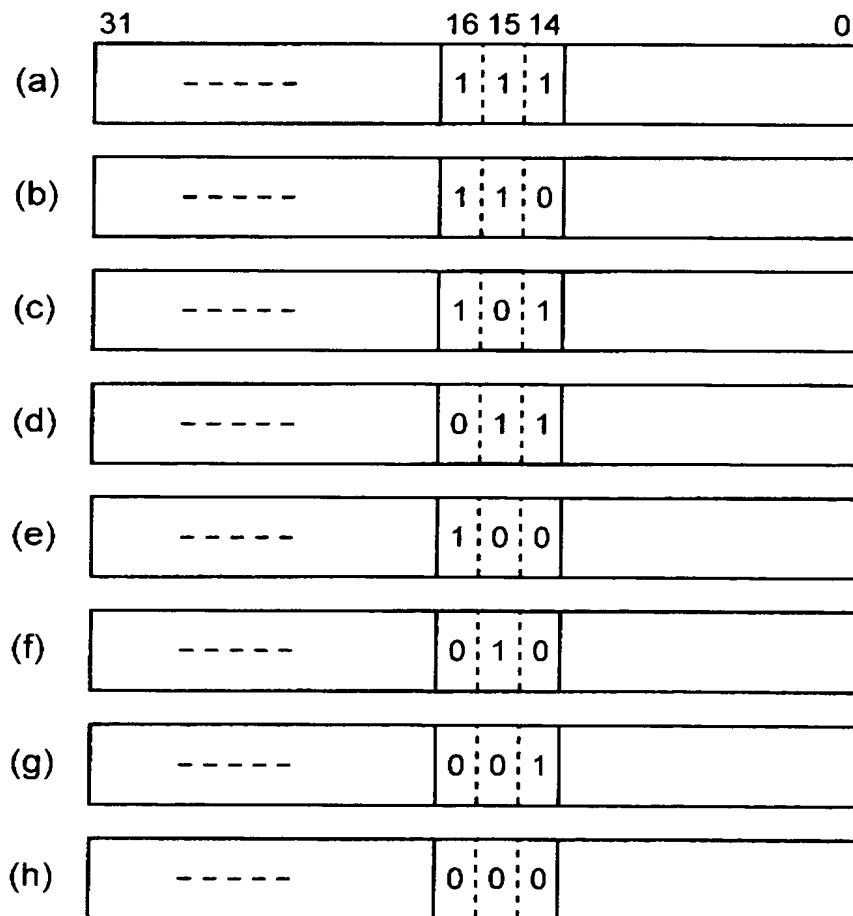
9サイクル

 : 命令配置済み

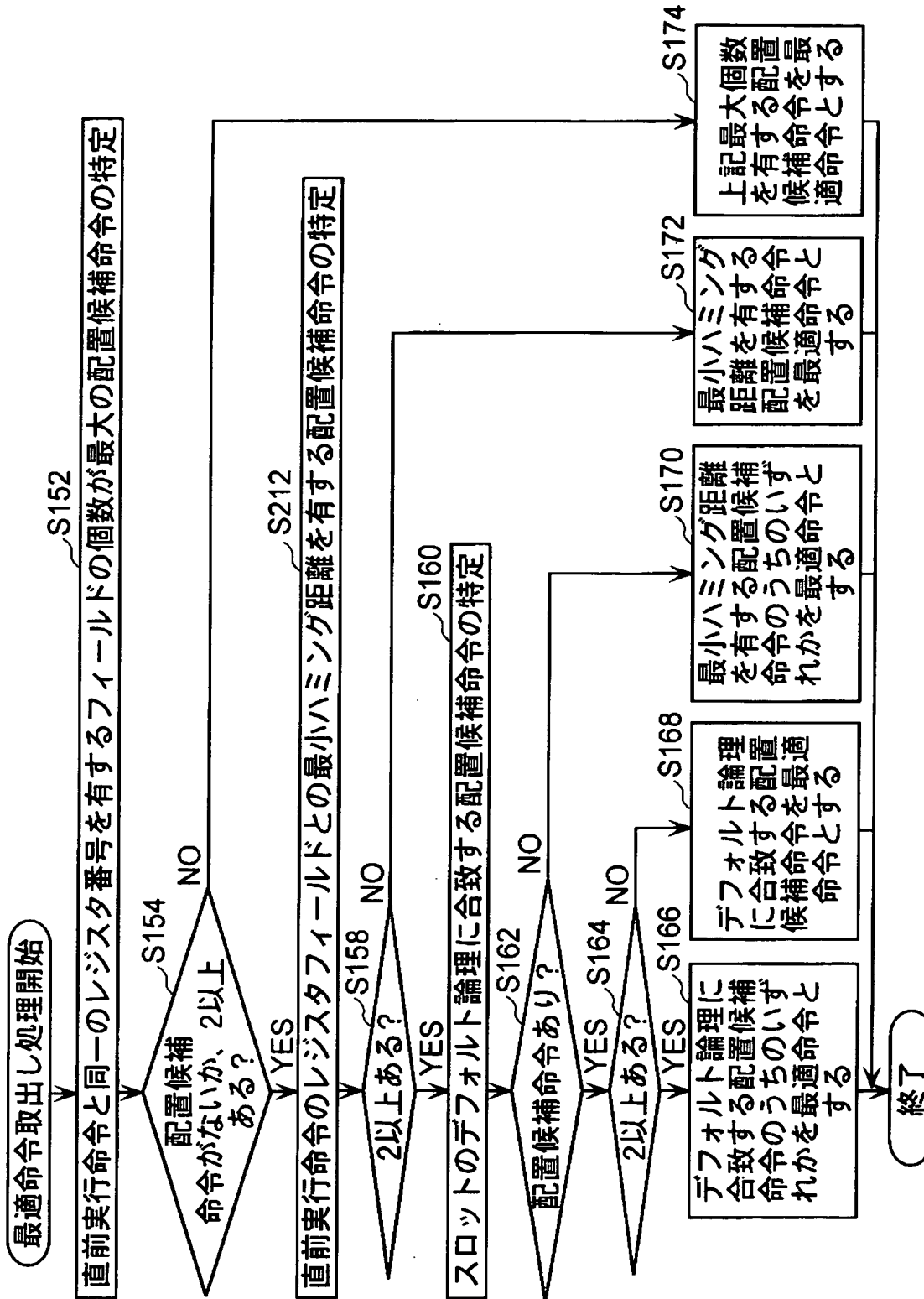
【図 37】



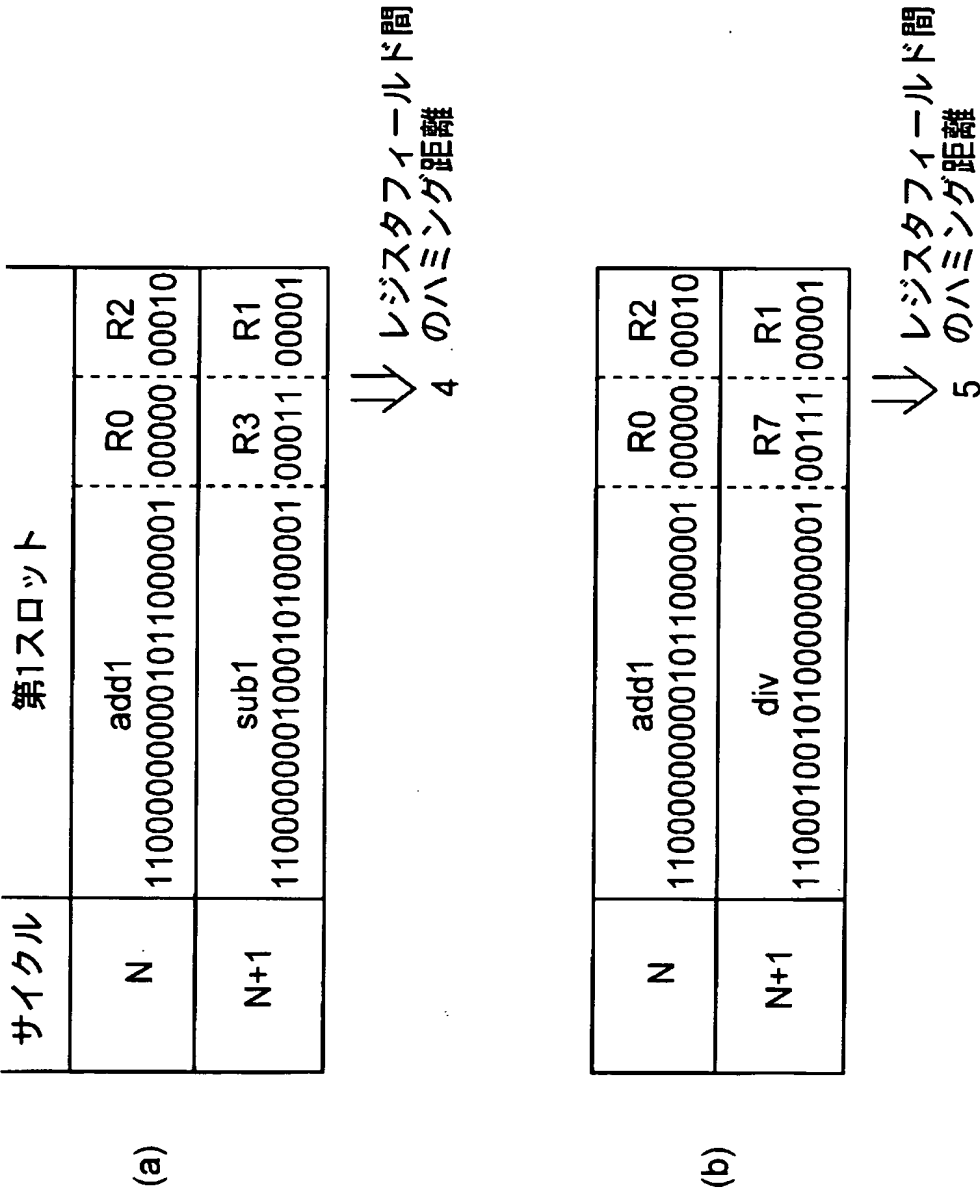
【図 38】



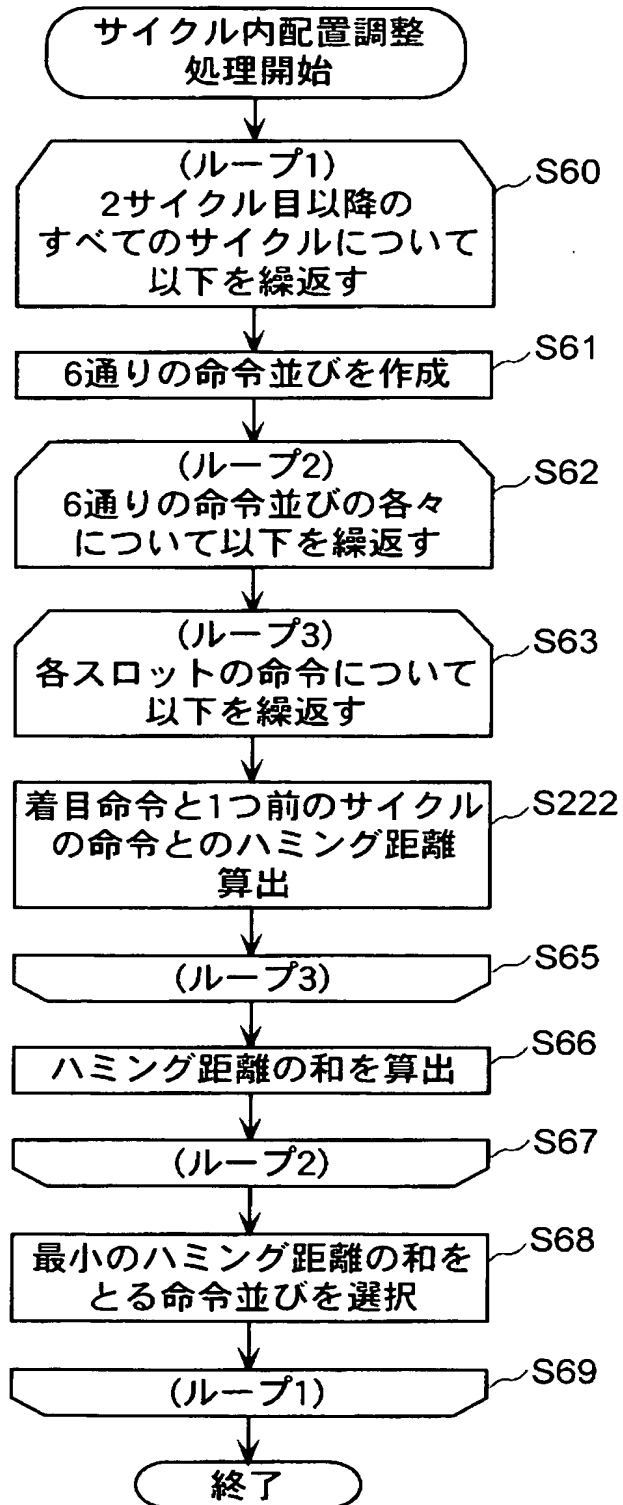
【図39】



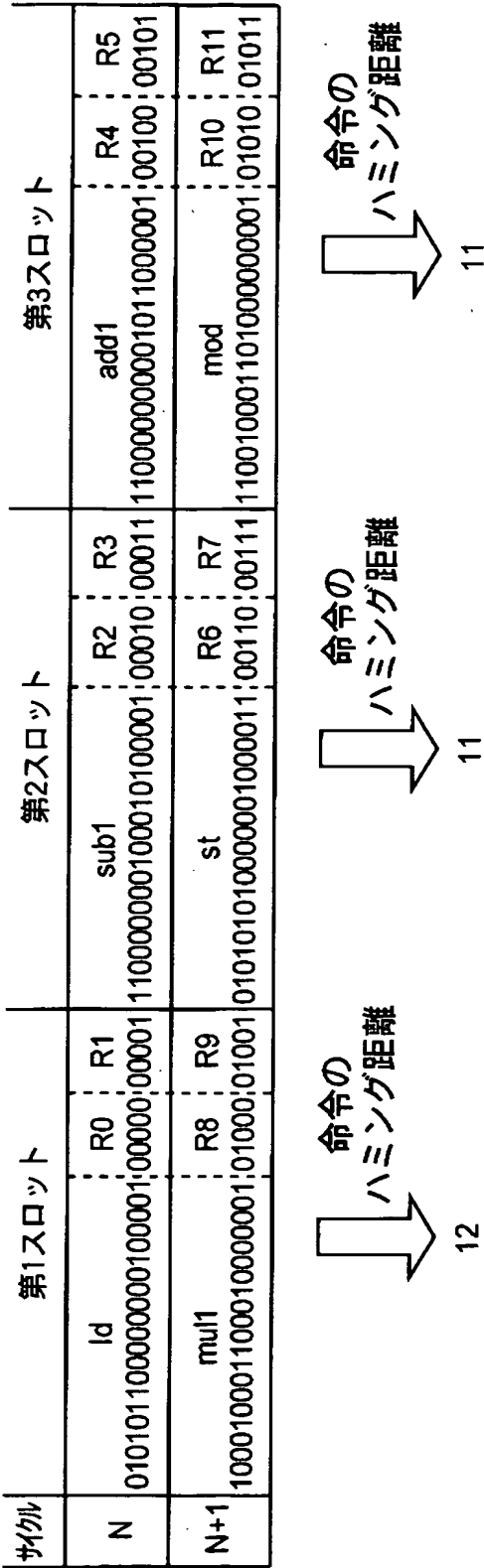
【図40】



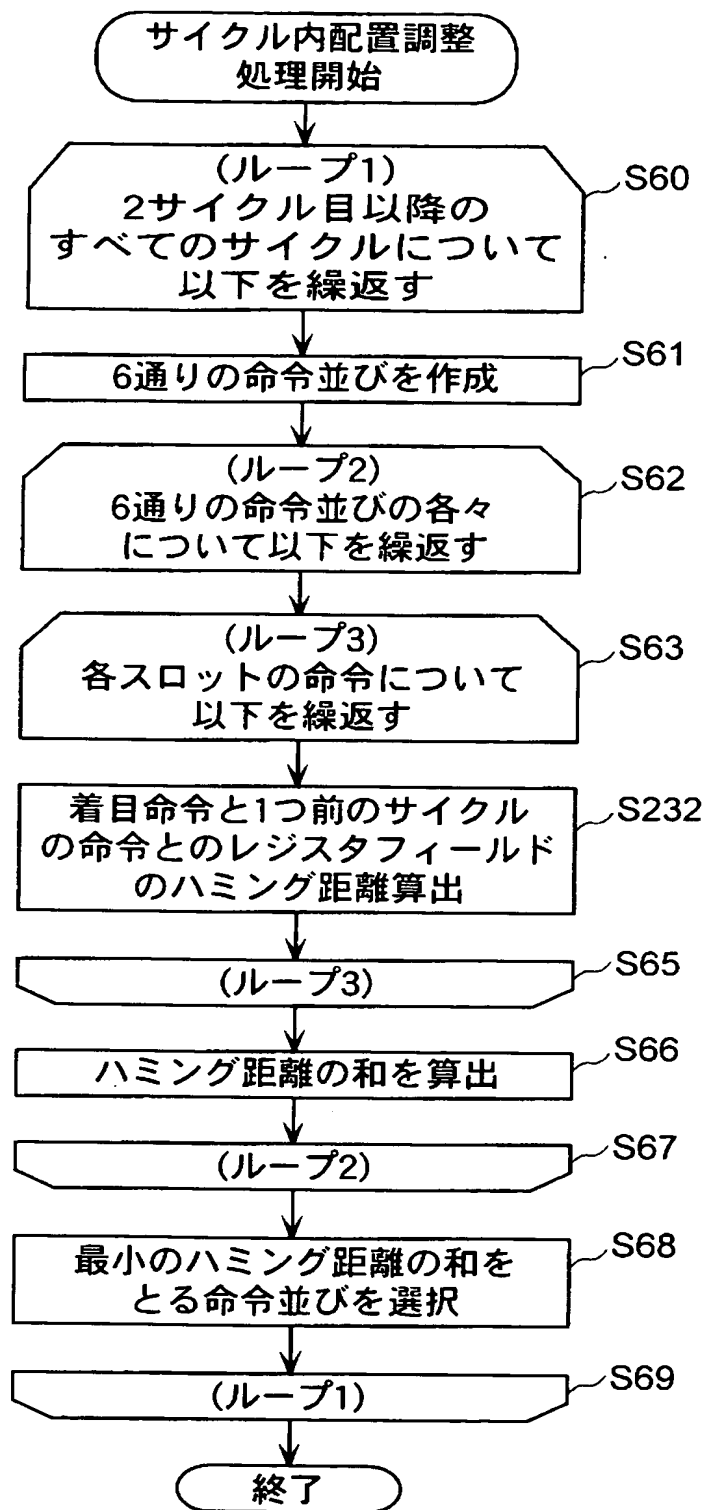
【図 41】



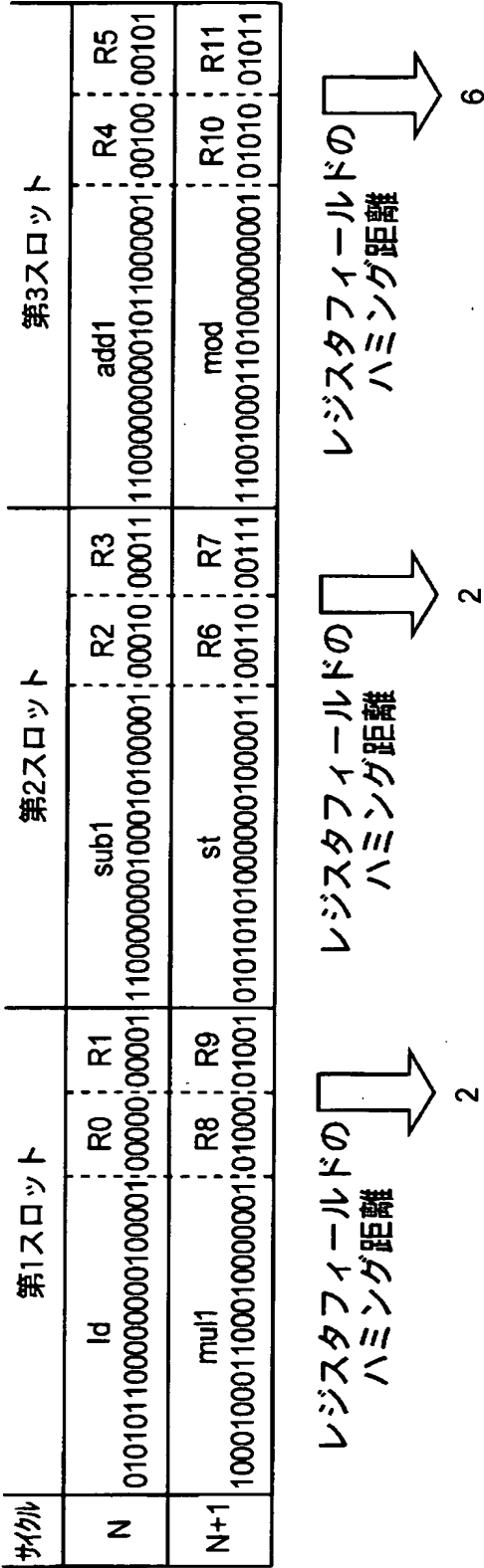
【図 4 2】



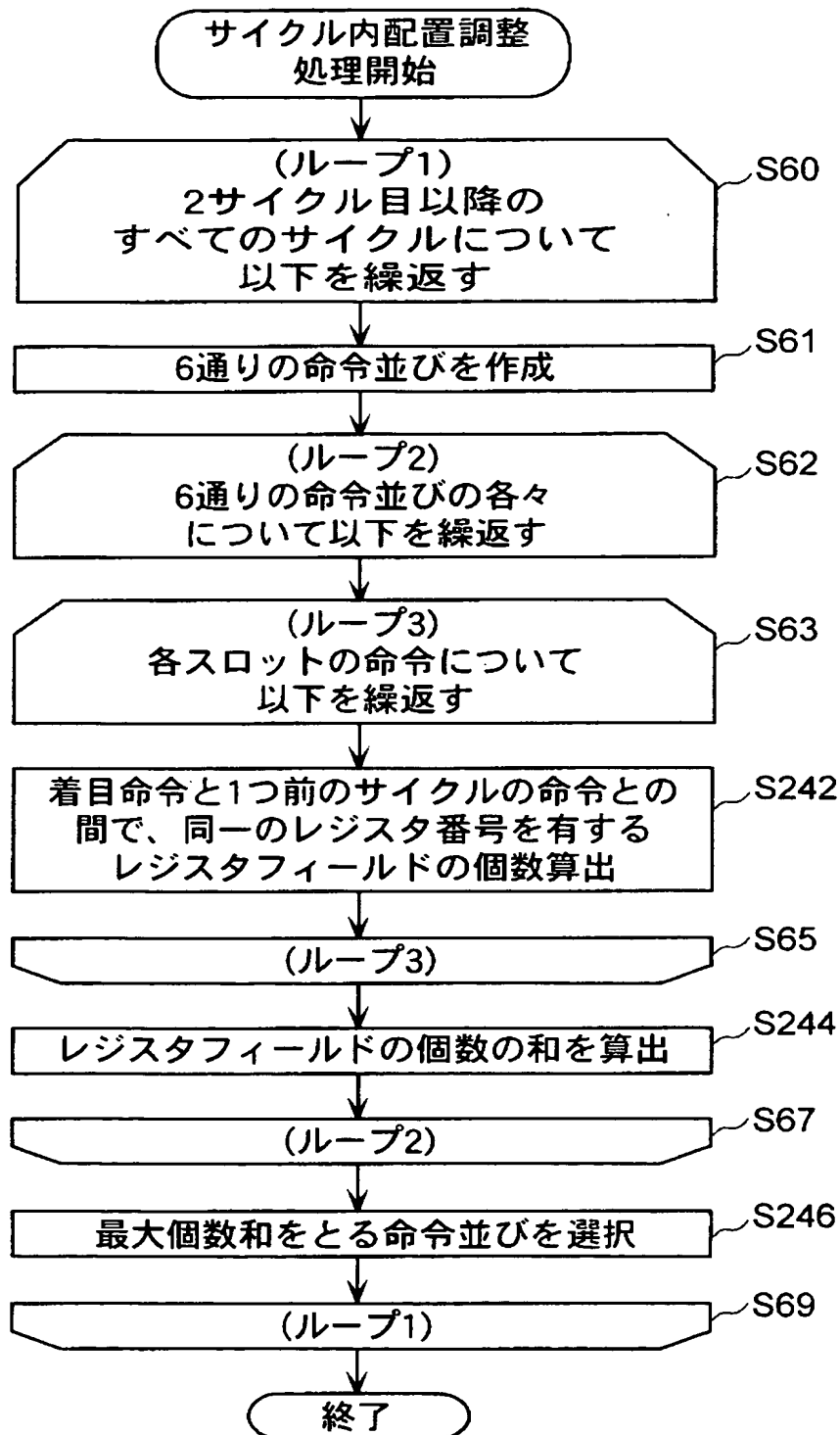
【図 43】



【図 4 4】



【図 45】



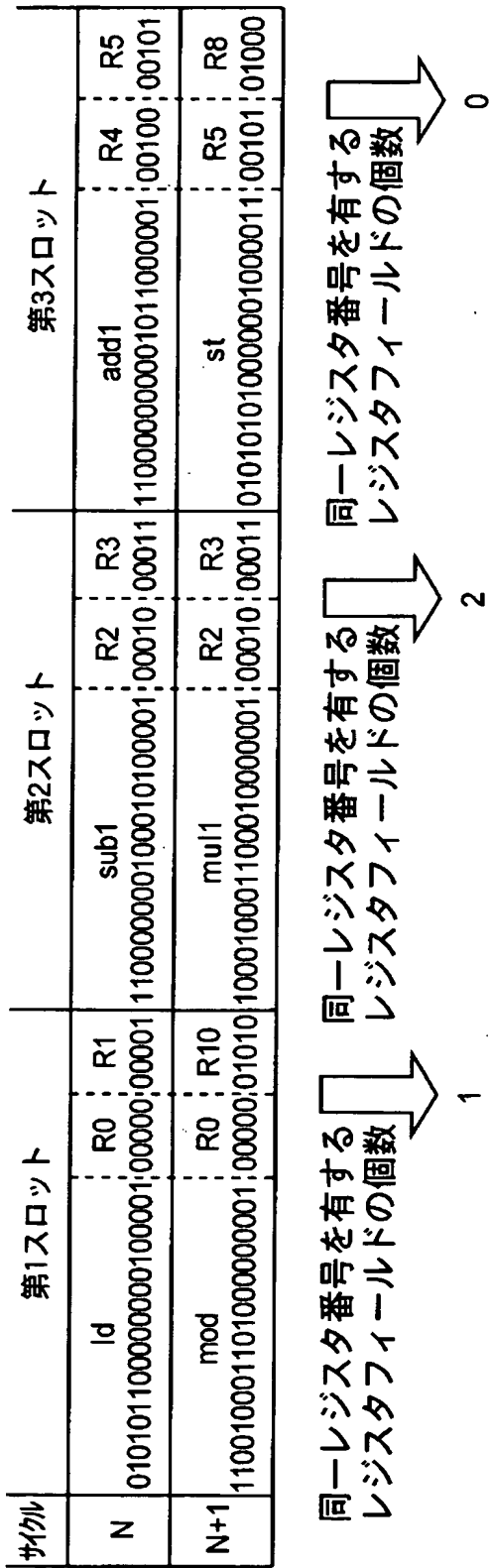
【図 4 6】

サイクル	第1スロット	第2スロット	第3スロット
- - -	- - -	- - -	- - -
N	Id 010101100000000100001:00000:00001 R0 : R1 00000:00001	sub1 110000000100010100001:00010:00011 R2 : R3 00010:00011	add1 1100000000001011000001:00100:00101 R4 : R5 00100:00101
N+1	st 0101010000001000011:00101:01000 R5 : R8 00101:01000	mul1 100010001100010000001:00010:00011 R2 : R3 00010:00011	mod 110010001101000000001:00000:01010 R0 : R10 00000:01010
- - -	- - -	- - -	- - -

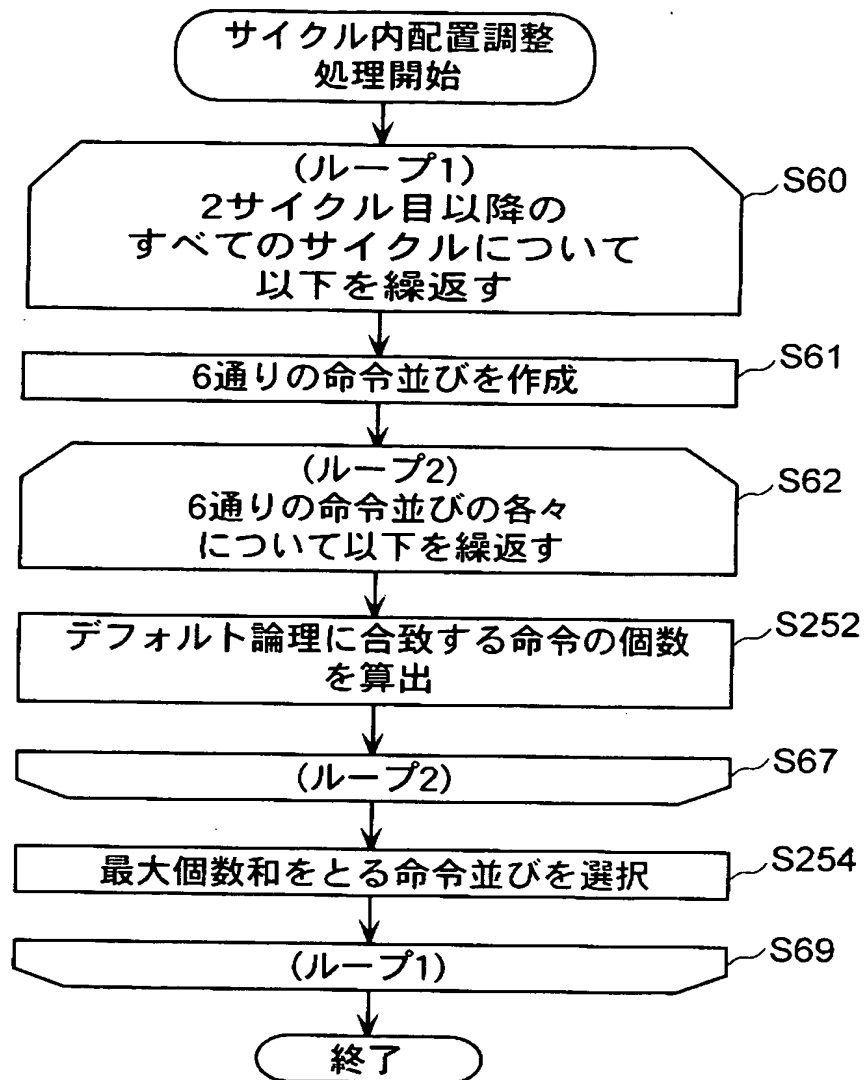
【図 47】

(a)	第1スロット	第2スロット	第3スロット
サクル	st	mul1	mod
N+1	010101010000001000011:00101:01000	1000100011000100000001:00010:00011	1100100011010000000001:00000:01010
	R5 : R8	R2 : R3	R0 : R10
(b)	第1スロット	第2スロット	第3スロット
サクル	st	mod	mul1
N+1	010101010000001000011:00101:01000	1100100011010000000001:00000:01010	1000100011000100000001:00010:00011
	R5 : R8	R0 : R10	R2 : R3
(c)	第1スロット	第2スロット	第3スロット
サクル	mul1	st	mod
N+1	1000100011000100000001:00010:00011	1010101010000001000001:00101:01000	1100100011010000000001:00000:01010
	R2 : R3	R5 : R8	R0 : R10
(d)	第1スロット	第2スロット	第3スロット
サクル	mul1	mod	st
N+1	1000100011000100000001:00010:00011	1100100011010000000001:00000:01010	0101010100000001000001:00101:01000
	R2 : R3	R0 : R10	R5 : R8
(e)	第1スロット	第2スロット	第3スロット
サクル	mod	st	mul1
N+1	1100100011010000000001:00000:01010	0101010100000001000001:00101:01000	1000100011000100000001:00010:00011
	R0 : R10	R5 : R8	R2 : R3
(f)	第1スロット	第2スロット	第3スロット
サクル	mod	mul1	st
N+1	1100100011010000000001:00000:01010	1000100011000100000001:00010:00011	0101010100000001000001:00101:01000
	R0 : R10	R2 : R3	R5 : R8

【図 48】



【図 4 9】



【書類名】 要約書

【要約】

【課題】 並列処理可能なプロセッサを低消費電力で動作させることができる命令列を生成可能なコンパイラを提供する。

【解決手段】 着目しているサイクルの3つの命令について並べ替えを行ない、6通りの命令並びを作成する（S61）。各命令並びの各スロットについて、着目命令と1つ前のサイクルの命令とのオペコードのビットパターン間のハミング距離を求める（S64）。その処理を3つのスロットの命令のすべてについて行ない（S63～S65）、ハミング距離の和を求める（S66）。以上の処理を、6通りの命令並びのすべてについて行なう（S62～S67）。6通りハミング距離の和のうち最小値をとる命令並びを選択し、その命令並びの並びになるように命令の並べ替えを行なう（S68）。以上の処理を、2サイクル目から最終サイクルまで繰り返す（S60～S69）。

【選択図】 図19

認定・付加情報

特許出願の番号	特願 2 0 0 3 - 0 1 9 3 6 5
受付番号	5 0 3 0 0 1 3 5 0 5 5
書類名	特許願
担当官	第七担当上席 0 0 9 6
作成日	平成 1 5 年 1 月 2 9 日

< 認定情報・付加情報 >

【提出日】 平成15年 1月28日

次頁無

特願 2 0 0 3 - 0 1 9 3 6 5

出 願 人 履 歷 情 報

識別番号

[0 0 0 0 0 5 8 2 1]

1. 変更年月日

1 9 9 0 年 8 月 2 8 日

[変更理由]

新規登録

住 所

大阪府門真市大字門真 1 0 0 6 番地

氏 名

松下電器産業株式会社